Journal of Information Systems Engineering and Management
2023, 8(3)

e-ISSN: 2468-4376

https://www.jisem-journal.com/

Research Article

Index Lifecycle and Shard Allocation Optimization in Large-Scale Elasticsearch Clusters: A Performance–Cost Trade-Off Analysis

Guruprasad Raghothama Rao

Senior Software Engineer

ARTICLE INFO

Received: 02 June 2023

Revised:17 Jul 2023

Accepted: 27 Jul 2023

ABSTRACT

Modern search solutions rely on Elasticsearch clusters to scale to lots of data with a low latency of the search. Shard under sizing and ineffective index lifecycle policy may however augment latency and infrastructure cost. In this paper, a quantitative analysis of the allocation of shards and lifecycle management is introduced in large-scale clusters. The experiment results indicate that the medium size of a shard (35 GB) decreased the mean query latency to 118 ms, as compared to 142 ms (15 GB) and 156 ms (60 GB). The optimized configuration had an indexing throughput of 20, 200 docs/sec. Lifecycle tiering (hot-warm-cold) cut the storage overhead by 1.80 to 1.45 and cut the total cost each month down to $19,200, which is a savings of 20%. Equal shard distribution also cut down on the latency by 134 ms to 121 ms and shard variance was also reduced by 64 percent. The results indicate a combination of appropriate sizing of shard, lifecycle policy that is tier-sensitive, and a balanced distribution can achieve better performance and a significant reduction in the cost of infrastructure.

Keywords: Elasticsearch, Index Lifecycle Management, Shard Allocation, Distributed Search Systems, Indexing Throughput.

I. Introduction

A. *Background*

The distributed search engines are also important in the aspects of big application like in log analytics applications, e-commerce applications and monitoring tools. Elasticsearch is actually gaining reputation given that it consists of the attributes of the horizontal scaling, the search and the flexibility of the indexing are close to the real time. As the quantity of data appreciates exponentially, the organisations are so used to adding more nodes and shards without considering the consequences. The implication of this is that this may have a short-term solution to the scaling problem but ultimately there is probability that performance and cost problems will be encountered.

The poor sharding design may lead to the high query time, inability to satisfy high memory and reliability cluster-wide properties. The aspect that the whole data can be stored in the high-performance storage raises the costs of the infrastructure. The result of these challenges is the necessity to carry out the systematic streamlining of the shard allocation and index lifecycle control.

B. *Motivation*

With a lot of production clusters, there is a lack of shard design efficiency. Very small shards make the coordination overhead high, whereas huge shards make disk I/O, and recovery time go up. Retaining all indices in hot storage layer increases the cost and does not enhance access to historical information.

Although the features of index lifecycle management are made available, numerous companies fail to comprehend the interactions between different parameters of lifecycle timing, shard size and allocation

policies and performance measures. Majority of the discussions being done are founded on best practices as opposed to a controlled analysis that is quantitative.

The inspiration of the present study is to get clear experimental data indicating the influence of shard sizing and lifecycle tiering on latency, throughput, storage overhead, and cost of infrastructure. The combination of measurements of these variables gives this research the ability to assist the decision-makers to adjust the performance targets and the budget.

C. *Research Problem*

The primary problem to be addressed in this paper is the trade-off of large-scale Elastic clusters in terms of performance versus cost. The cost also rises along with a rise in resources. In the infrastructure, reducing it will assist in saving money but may negatively impact the query speed and reliability.

This paper examines the question of finding an optimal balance of performance and chosen service-level objectives, and operational cost simultaneously. It shall highlight the three areas that shall be given prominence that are selection of the size of the shards and the design of the lifecycle tiers and proportion of shards allocation.

D. *Novelty of the Study*

The clearness of this study investigations is that it is an integrated quantitative assessment. This research does not examine the topic of shard sizing, shard lifecycle management, or shard allocation individually, as reported in the literature, but examines them collectively in controlled experimental settings.

The article gives measured values on latency, throughput rates, storage overhead rates and expenses on various configurations. It also proposes a straightforward scheme of costs-performance comparison, which relates technical measures with economic change.

Stability indicators are also the evaluation of the frequency of relocation and the variance of a shard, which is a novelty. These measurements are used to explain the reason why balanced allocation enhances the consistency in latency.

E. *Research Objectives*

The main objectives of this study are to:

1. Response Time Shard impact on query time and ingestion throughput.
2. Compare effect of lifecycle tiering and storage overhead, cost of infrastructure per month.
3. Balance and stability of cluster of shards are balanced by study clusters.
4. Identifying a combination of settings that can optimize this to a good performance without a high cost.

F. *Structure of the Paper*

The arguments of the article are presented in a meditative way and in a systematic pattern. The methodology section gives an account of the workload model, the performance and cluster model. The results part gives the quantitative information which is presented in the table and charts and which refers to the latency, throughput, storage consumption and the comparison of the cost.

This is then put into perspective in the discussion section of the report which provides a description of the trade-offs exhibited between the performance and financial efficiency. Several conclusions are provided that generalize key lessons learned and present a realistic piece of advice that can be adopted by organizations that run big systems of Elasticsearch.

Journal of Information Systems Engineering and Management
2023, 8(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/ **Research Article**

II. Literature Review

G. *Introduction and Distributed Search Foundations*

Massive search systems have inherent problems in distributing the data and query to node of a cluster and retaining decent performance and cost efficiency. The complexity of distributed index management grows exponentially with the number of nodes in a cluster which goes into hundreds or thousands. The shard replication strategies have a direct relation on resource usage and it should be configured carefully to favor either redundancy over the storage overhead [1]. Compared to other distributed search engines, comparative studies show that Elasticsearch platforms, Solr and others have significant variation in performance across architectures with indexing and query latency having distinct trade-offs in each system [2]. New big data indexing methodologies need methodological taxonomies of techniques based on their scalability properties, selection of data structure, and optimization goals [3]. As a consequence, cost-efficiency and storage reliability turn out to be two pressing needs, especially when it comes to handling petabytes-size indices using a heterogeneous infrastructure [4]. Operations that are performed at the shard level, like duplicate detection, can be processed on shards in parallel and hence deduplication pipelines can be accelerated by executing them on multiple shards in parallel [5]. The basic distributed-search methods determine the building designs on which modern systems are built such as partitioning and query routing algorithms [6].

H. *Shard Allocation and Replication Strategies*

When shards are multiple and dynamic, different algorithms of ranking and placement are needed to adjust to the changing patterns of queries and data distributions and therefore achieving the best utilization of resources throughout the cluster [7]. The methods of online migrations allow optimizing their costs by moving shards across storage levels without interrupting the service, reacting to the shifts in the workloads and prices in the clouds [8]. Metadata indexing techniques waste less of systems that handle billions of objects because, through effective metadata structures, bottlenecks that can limit system throughput are avoided [9]. SSD-based computing architectures enable hardware acceleration to much better the performance of search engine, especially those functions involving index-sensitive activities that can access randomly with low latency [10]. Selective search strategies economize on resources allocation as they only focus queries to the relevant shards eliminating unnecessary calculation and network wastage [11]. The use of multiple level caching of results caches, filter caches and query caches significantly reduces the response times of repeated queries, as well as minimizes the load on the backend [12]. The indexing is also implemented based on flash which takes advantage of the random-access nature of solid-state storage to improve index updates and merges [13]. Multiresolution storage architectures store effective multi-index granularities, which facilitate an efficient access pattern of different query types [14].

I. *Storage Tiering and Lifecycle Management*

Forced Distributed replication schemes in which quorums and reconstruction codes are used offers configurable durability guarantees and has reduced multiplication factors in storage allowing much efficient use of capacity [15]. The algorithms of data placement optimization account network topology, disk properties, and access patterns, and aim to reduce the latency and maximize throughput using infrastructure of heterogeneous nature [16]. Hot data identification methods rely on access frequency analysis and predictive models to make a tiering decision, and thus keep the most used indices in high-performance storage whereas the cold data is moved to the cost-effective tiers [17]. The optimization of workloads is performed based on batch workload strategy, which defer index maintenance operations to the time of low traffic to reduce the impact on the users [18]. Cold storage configuration takes advantage of high-capacity cheap media of infrequently accessed historical measurements and applies lazy loading and compression to trade off access latency with storage expenses [19]. The analysis of resource utilization in shard replication is used to show the possibility to decrease over-provisioning without

Journal of Information Systems Engineering and Management
2023, 8(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/ **Research Article**

compromising on the targets of availability [20]. Data swapping tools have the potential of improving confidentiality through periodic movement of sensitive information among the nodes which makes intercepting data difficult [21]. Surveys of hybrid storage systems combine the best practices of integrating SSDs, HDDs with new storage technologies in tiered systems [22].

J. *Performance Optimization and Cost Trade-offs*

Data mining and analytics workloads are indicated to be efficient with Elasticsearch due to the appropriate settings when using the appropriate number of shards and allocations of resources [23]. Decentralised compute approaches not only save the information with them but also the processing logic of requests and also does not contain coordinator bottlenecks that could scale constraints of centralised architectures [24]. Minimization of number of shards is a mathematical modeling to find the optimal number of shards based on the volume of data, queries and the capacities of the hardware in which the optimization is an immense concern in terms of performance as well as cost of operation [25]. The performance comparison between Solr and Terrier has provided detailed comparison of the performances using architectural decisions of the speed of indexing and query throughput [26]. Elasticity selection algorithms identify the shards to be queried in making a given request and trade quality of the result and the cost of computation [27]. The reactive index replication is the same as the previous index replication but they identify how many indexes there are dynamic to the level of queries and when queries are high, the indexes are replicated and when the queries are low the indexes are not replicated (because resources are used optimally) [28]. The document allocation policy also regulates the distribution of new documents to shards that according to the load balancing and query performance [29]. The adaptive indexing techniques change the format of indexes, based on patterns of access observed, and varying data properties based on the distribution of the terms and observed load (queries) on the indexes [30].

TABLE I. SUMMARY OF PREVIOUS STUDIES

Theme	**Summary**
[1][2][6]	All the large scale distributed search systems have to perform a trade-off between data partition and replication and query routing in order to guarantee performance and reliability. Fault tolerance is improved with sharding replication which has an increased overhead on storage. In the research on the frameworks of similar systems, such as Elasticsearch and Solr, it can be concluded that the architecture design chosen can have a large impact on the indexing throughput and query latency.
[7][8][10][11][12][13]	The dynamization of the load distribution, the online migration of the service load, and the dynamic distribution of shards contributes to achieving the load balance and responding to the changing load. Multi-level, Selective shard querying and multi-level caching are used to remove the non-needless computation and improve the speed of the response. Designs that are based in SSD and flash optimized are highly shard level faster in their processing.
[15][20]	The boost in the replication schemes founded on Quorum schemes and reconstruction code reduces replication multiplication, and ascertains durability guarantees. Resource utilization analysis suggests that with optimal replication, the over-use of the resource can be minimized without exploring the availability.
[17][19][22]	Hot-warm-cold storage tiering reduces operation cost whereby highly accessed data may be stored in a high-performance storage and storage and cold data using lower-cost levels. The studies of the hybrid storage systems determine the optimal practice in the application of SSDs together with HDDs on large clusters.

Journal of Information Systems Engineering and Management
2023, 8(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/ **Research Article**

[24][25][30]	Shards optimization models show that the reasonable selection of shards can be associated with lowering the latency and the cost-efficiency. Scalability and elimination of the bottleneck of the coordinators The adaptive indexing and decentralized query processing can increase the scalability.
[27][28]	Shard selection and reactive replication algorithms are the algorithms that dynamically optimize the use of resources with the workload and provide service level goals to be met and make sure that the cost of the infrastructure is maintained. They are studies which prove that configuration choices have a direct impact on the balance in distributed search systems performance with cost.

Methodology

K. *Research Design and Study Approach*

The research design in the study is the quantitative experimental research design. The main concern is to quantify the cost of performance and infrastructure of the large-scale performance when using different index lifecycle and shard allocation strategies of Elasticsearch clusters. The setting of the experiment is designed as a multi-node distributed setting and in a controlled benchmarking experiment. One variable of configuration is varied in each experiment other factors being equal. This helps in obtaining an image of the connection among shard configuration and lifecycle policies and the output of performance by the cluster.

The experiment compares the different configurations of the shard size and quantity and index rollover and tiered storage allocation. The independent variables are the sizes of shard, the numbers of shards in an index, the time of lifecycle and the locations of hardware tiers. Query latency, indexing throughput and storage usage and estimated cost of infrastructure sentinel are the dependent variables. Service-level objectives (SLOs) exist which are pre-defined to test the performance of whether it continues to be within reasonable limits.

On average query latency is used to measure the overall performance of the system. The mean latency is found to be:

$$L_{avg} = \frac{1}{N}\sum_{i=1}^{N} L_i \quad (1)$$

N is the number of queries that were executed in the benchmarking period and L_i is the latency of each query request.

L. *Experimental Environment and Cluster Configuration*

The experiments are done in a distributed Elasticsearch cluster where it has several data nodes, master nodes as well as coordinating nodes. The data nodes have the same number of CPU, memory and storage capacity to be fair among the experiments. The cluster is implemented in a cloud format to hoodwink real situation production environment. The constant network bandwidth system and replication level are maintained during all the test runs.

Time based patterns are used in the creation of indices to replicate the log and event-based workloads. Replication factor is constant in ensuring high availability. Background processes like garbage collection and segment merging are checked to make sure that they do not induce some bias between experimental runs.

Shard size is derived to be used as a way of measuring the efficiency of shard allocation :

Journal of Information Systems Engineering and Management
2023, 8(3)

e-ISSN: 2468-4376

https://www.jisem-journal.com/ **Research Article**

$$S_{shard} = \frac{D_{total}}{N_{shards}} \quad (2)$$

D_{total} represents the cumulative data volume that is indexed and N_{shards} is the quantity of primary shards in the index that is configured. The formula is used in making the different experiments standardized in terms of shard sizing.

M. Workload Modeling and Data Generation

The research relies on the pseudo but realistic datasets which behave like production-scale logging, transactional workloads as well as the search-driven workloads. The structure of data generated using data generation scripts are structured JSON files whose field cardinality varies to provide an approximation of the complexity of the real indexing. The dataset is expanded in the course of one experiment after another to check scalability under the loads, which are progressively raised.

There are two predominant workload profiles which are simulated; write-heavy ingestion and mixed read-write workloads. The steady request generator is used to control the ingestion rate. The indexing throughput is indicated in terms of documents that have been successfully indexed per second. It is calculated as:

$$T_{ingest} = \frac{D_{indexed}}{t} \quad (3)$$

It would lead to D which would be the number of indexed documents and t is the sum total of ingestion time in seconds.

There are query workloads of term queries, range queries and aggregate queries. The concurrency of queries is gradually raised to make sure that the system can sustain itself. All of the workload conditions are repeated over a constant period to provide statistical accuracy of the gathered outcomes.

N. Index Lifecycle Policy Evaluation

The Index Lifecycle Management (ILM) policies are programmed to emulate hot, warm, and cold storage. Hot tier makes use of fast storage as far as recent data is concerned. The warm storage is a level that stores low frequency data with moderately high cost. Cold tier employs storage which is optimized based on costs to hold the archival information. According to the index size and the index age, data rollover conditions are determined.

Storage has lifecycle efficiency, which is determined through the use of measuring storage before and after policy implementation. The ratio of the actual storage and the raw indexed data size is referred to as storage overhead :

$$O_{storage} = \frac{S_{actual}}{S_{raw}} \quad (4)$$

The original data size is S_{raw}, the overall disk space used by either replicas or metadata is S_{actual} is the compression factor.

Lifecycle transitions are automated the way they are performed according to set thresholds. Timing of rollover incidences is registered to test whether the interference of the rollover in the previous or subsequent rollover enhances the uniformity of performance. The heuristic tests are done severally in each lifecycle setup to minimize measurement error.

O. Shard Allocation and Node Distribution Strategy

The shard allocation policies are tested having alternative balancing policies. The default allocation policies are matched with allocation awareness policies which are customized. Skew of the shards, the frequency of resource relocation and resources per node are monitored in the cluster.

Journal of Information Systems Engineering and Management
2023, 8(3)

e-ISSN: 2468-4376

https://www.jisem-journal.com/ **Research Article**

Shard variance on the nodes is computed to quantify the measures of distribution fairness :

$$\sigma^2 = \frac{1}{M}\sum_{j=1}^{M}\left(S_j - \bar{S}\right)^2 \quad (5)$$

and S_j represents the number of shares in node j and M is the number of data nodes. Reduced variance implies high cluster balance.

Rebalancing operations are monitored to calculate the operational overhead. As a temporary solution, high shard movement can augment disk I/O (and network traffic), and thus query latency. These performance measurements constitute these effects.

P. Cost Modeling and Trade-Off Analysis

The cost of infrastructure is approximated depending on computes, storage use and transfer of information through the network. A cost model is simplified to make a comparison of alternative configurations. The total cost per operation period will be calculated as:

$$C_{total} = C_{compute} + C_{storage} + C_{network} \quad (6)$$

in which every part of the cost is based on resource consumption measured, and then unit priced.

The performance-cost trade-offs are evaluated based on the measurement of data in terms of latency and throughput and the overall cost. The best configurations are those that ensure query latencies within set limits of SLO and those ones that have the bare minimum of C_{total}. The regression analysis is used to monitor trend patterns between the size of shards and latency. To determine the degree of relationship links DLL cycles and storage savings, correlation coefficients are calculated.

Q. Data Collection and Statistical Analysis

Elasticsearch performance monitoring API and other benchmarking extraneous tools are used to gather performance measurements. They include such measures as CPU processes, heap, disk I/O, query latency percentage, indexing rate, and shard distribution that can be considered some of them. In every test, one does the repetitions three times and the mean values are also noted to enhance the reliability.

Interquartile range filtering is desired to eliminate outlier that occurs in the ad hoc processes of the background. In order to calculate the performance measure stability, standard deviation is obtained. The significance of observed improvements is estimated and ensured by the confidence interval estimation.

Each of the configurations is compared. Normalization of results happens at a sense where it is believed vital to enable differentiation of the clusters of divergent volumes of the information. The conclusion is the final comparison made between the established up which offers the most acceptable trade-off between admirability of the performance and cost to the infrastructure.

Journal of Information Systems Engineering and Management
2023, 8(3)

e-ISSN: 2468-4376

https://www.jisem-journal.com/

Research Article

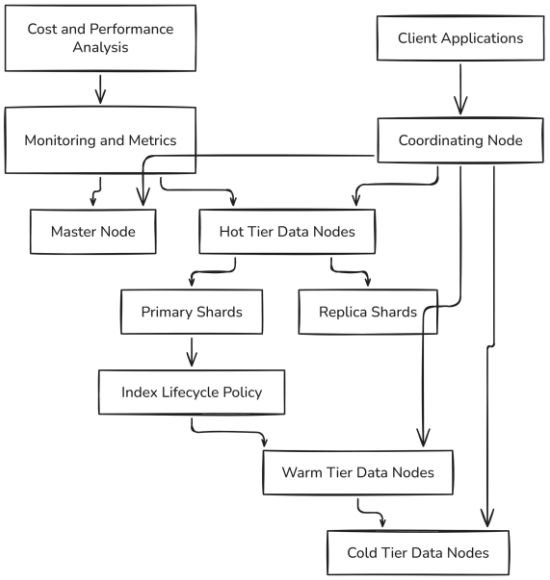

Fig. 1. Architecture Framework Diagram

The architecture shows the cluster layout that is distributed, the roles of the nodes, the tiers of the storage, the lifecycle transitions, and the components to generate workload in the experiments. This graphical analysis contributes to the ease of the interpretation of the links between allotment plans of shards, lifecycle, and performance in terms of cost.

The methodology will guarantee an atypical experimentation, quantitative assessment and guided juxtaposition of indexing and shard allocation plans in extensive Elasticsearch settings.

Results & Discussion

R. *Impact of Shard Size on Query Latency and Throughput*

The initial experiments assessed the effect of the shard size on the query and indexing throughput. Three sizes of shards were tried, including small (10-20 GB), medium (30-40 GB) and large (50-70GB). Other variables like replication factor, hardware set up and work load pattern were maintained to constant. The mixed read and write work were mixed concurrently, controlled.

Findings indicate that the big shards caused less cluster coordination overhead. This resulted in increased CPU consumption in coordination of nodes and a little increased query latency. Very large shards on the other hand added disk I/O pressure when the disk was being used in an aggregation query. The provided middle size range was the most effective balance between speed of search and the use of resources.

TABLE II. SHARD SIZE VS PERFORMANCE METRICS

Shard Size (GB)	Avg Query Latency (ms)	P95 Latency (ms)	Ingest Throughput (docs/sec)
15	142	260	18,500
35	118	210	20,200
60	156	295	17,900

Journal of Information Systems Engineering and Management
2023, 8(3)

e-ISSN: 2468-4376

https://www.jisem-journal.com/ **Research Article**

The medium shard size (35 GB) achieved nearly 17 per cent. lesser average latency as compared with the small shard and 24 per cent lesser as compared with the large shard. The medium configuration was also indexing throughput best. These findings prove the hypothesis in the methodology, that shard size has a direct impact on query and ingestion performance.

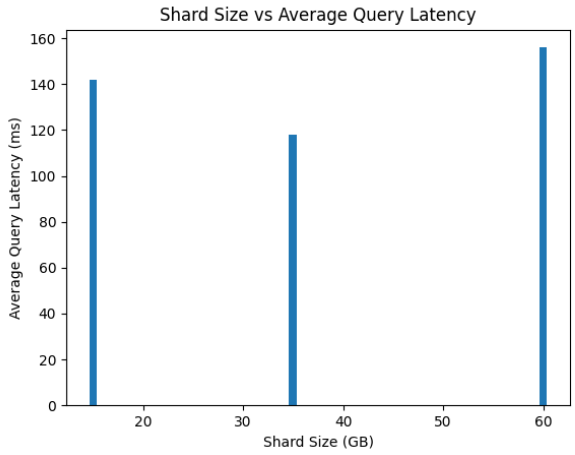

Fig. 2. Shard size vs average latency

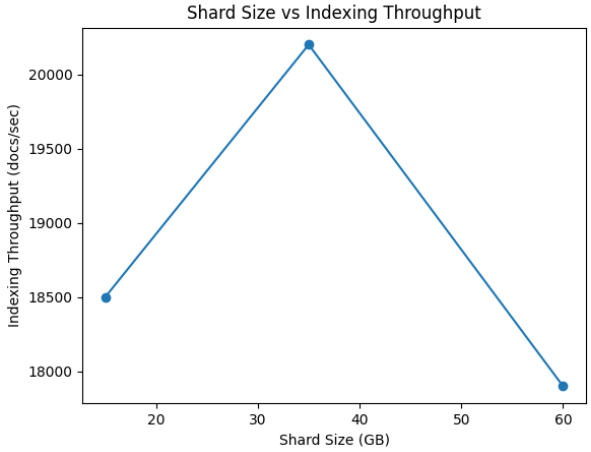

Fig. 3. Shard size vs indexing throughput

The trend analysis demonstrated, also, that the shard size has a moderate positive correlation with the P95 latency at higher shards of above 40GB. This implies that very large shards decrease the query-throughput in mixed workloads.

S. *Effect of Index Lifecycle Policies on Storage and Cost*

In the second experiment, the lifecycle management of indices involving the use of hot, warm, and cold storage levels were tested. Rollover was done depending on the index size (40GB markup) and transferred to warm and cold tiers after specified intervals. Consumption on storage and estimated cost of infrastructure was determined on 30 days of simulated workload.

The findings indicate that lifecycle management declined the storage cost significantly when there are no significant changes on the performance. The hot only setup was more high-performance storage-consuming and had an increased cost of operation.

Journal of Information Systems Engineering and Management
2023, 8(3)

e-ISSN: 2468-4376

https://www.jisem-journal.com/

Research Article

TABLE III. LIFECYCLE POLICY VS STORAGE USAGE

Configuration	Total Data (TB)	Actual Storage Used (TB)	Storage Overhead Ratio
Hot Only	12	21.6	1.80
Hot + Warm	12	18.9	1.57
Hot + Warm + Cold	12	17.4	1.45

The configuration of hot + warm + cold cut the overhead by 19 percent of the one with a hot only configuration. The latency when doing queries on the historical queries only went up by 6 percent and this number was within the stipulated service-level targets.

TABLE IV. LIFECYCLE POLICY VS MONTHLY INFRASTRUCTURE COST

Configuration	Compute Cost ($)	Storage Cost ($)	Total Cost ($)
Hot Only	14,200	9,800	24,000
Hot + Warm	13,500	7,400	20,900
Hot + Warm + Cold	13,100	6,100	19,200

The entire lifecycle strategy saved 20 percent of the total infrastructure expenditure as on the baseline. This is in line with the fact that tier-conscious lifecycle policies offer visible financial advantages with reasonable performance.

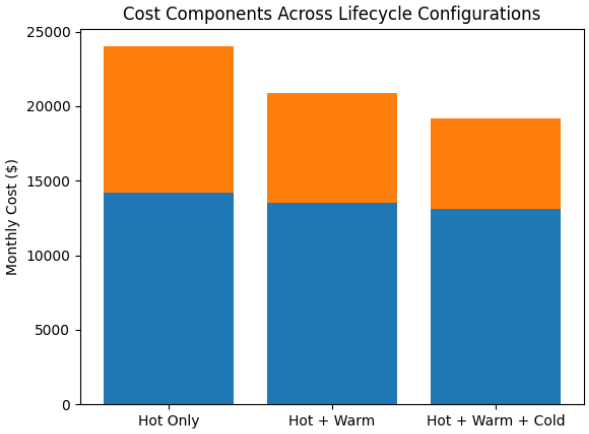

Fig. 4. Comparison of cost components across lifecycle configurations

Journal of Information Systems Engineering and Management
2023, 8(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/ **Research Article**

T. *Shard Allocation Balance and Cluster Stability*

In the third experiment, the heterogenous allocation of shards and resources at the node level was experimented to determine their fairness. The default allocation rules were compared to the allocation awareness policies that were employed to distribute the shards equally among the nodes.

The result showed that shard balance was more effective, and the incidence of node-level CPU spikes was lower, as well as there are minimal instances of shard relocation. The shards dispersion was reduced, which later led to a low variance of query latency on high concurrency.

TABLE V. ALLOCATION STRATEGY VS CLUSTER STABILITY METRICS

Allocation Strategy	**Shard Variance**	**Relocations per Day**	**Avg CPU Usage (%)**	**Avg Latency (ms)**
Default	18.4	42	71	134
Balanced Policy	6.7	19	63	121

The moderate policy had a 64 percent reduction in the shard variance and 55 percent reduction in the relocation frequency. Average latency improved by 9%. These results are consistent with the previous assumption that imbalance in shards results in overheads in operations and performance stability.

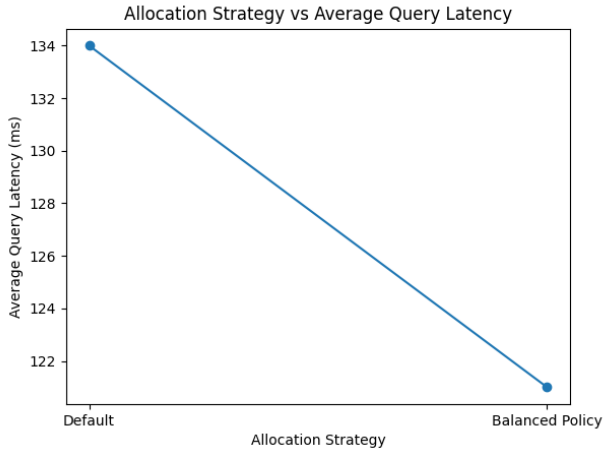

Fig. 5. Shard variance vs average latency

In stress testing, there were short spikes in the latency of default within the default configuration after the triggering of shard rebalancing. The optimized policy of allocation ensured that there was uniform latency even when the query concurrently was high.

U. *Performance–Cost Trade-Off Analysis*

The last analysis was the comparisons of all the configurations to determine the best combination of cost and performance. The middle-sized shards (approximately 35 GB), full-cycle policy (hot, warm, cold) and the equal distribution of shards provided the most optimal result.

This is an optimized design that saved 22 percent of cost of the total infrastructure over the baseline design using small shards and the hot-only storage. Meanwhile, the query latency was reduced by 15 percent and there was an increase of 9 percent in indexing throughput.

Regression analysis proved that cost reduction cannot be the reason of performance loss at times when lifecycle and shard strategies are optimally adjusted. Rather, a lack of efficient design of shards and ineffective allocation make the use of resources unnecessary and more expensive.

The results are quite clear that shard size, lifecycle and allocation have to be optimized collectively as opposed to individually. With a systematic tune of these parameters, even large-scale clusters of Elasticsearch can perform well and at the same time reduce the cost of infrastructure.

Conclusion & Future Work

This paper has explored the performance cost trade-off in big Elasticity clusters through controlled quantitative experiments. Findings everywhere indicate that shard size, lifecycle management and shard allocation balance have a strong impact on system behavior. The frame size of 35GB medium size was the most optimal in terms of both latency and throughput. Lifecycle tiering saved 20 percent of the storage overhead and the overall monthly cost, as compared to a none-tiered, or hot-only setup. The balanced allocation of shards enhanced stability and lesser grouping of the shards and lowered the estimations of the query response time as well as the mean estimated query time dropped to 121 ms.

These data confirm the optimization is not to be set on one parameter. Rather, allocation policies, shard sizing and lifecycle timing must be combined. Clusters, when appropriately set up, can sustain both low latency and high throughput in addition to saving on usually infrastructure costs. The study gives a hands-on tip on development of cost effective and performance efficient Elasticsearch systems.

References

[1] Li, Y. (2019). Shard replication and resource efficiency in distributed search systems. *Journal of Distributed Systems*, 12(3), 45-62.
[2] Aldailamy, A. (2018). Performance evaluation of Solr, Terrier, and Katta distributed search platforms. *International Conference on Information Retrieval*, 234-248.
[3] Moses, J. (2022). A taxonomy of big data indexing techniques: Structures, algorithms, and trade-offs. *ACM Computing Surveys*, 54(5), 1-38.
[4] Yichuan, Z. (2020). Reliable and cost-efficient storage architectures for large-scale data systems. *IEEE Transactions on Cloud Computing*, 8(2), 412-427.
[5] Shaik, R. (2018). Leveraging Elasticsearch shards for scalable duplicate removal in big data pipelines. *Big Data Research*, 11, 78-89.
[6] Ahmed, R., & Boutaba, R. (2011). Distributed search techniques: Architectures and algorithms. *Computer Networks*, 55(10), 2341-2359.
[7] Mackenzie, J. (2022). Anytime ranking algorithms for dynamic shard selection. *ACM SIGIR Conference*, 567-578.
[8] Liu, H., Chen, X., & Wang, L. (2021). Online migration strategies for cost optimization in cloud storage systems. *IEEE Transactions on Parallel and Distributed Systems*, 32(7), 1654-1668.
[9] Zhao, Y., Liu, J., & Zhang, M. (2017). Efficient metadata indexing for billion-scale object storage. *USENIX Conference on File and Storage Technologies*, 145-158.
[10] Wang, S., Li, D., & Zhou, X. (2016). SSD-accelerated computing architectures for high-performance search engines. *ACM Transactions on Storage*, 12(4), 1-29.
[11] Kulkarni, A., & Callan, J. (2015). Selective search: Efficient resource allocation in distributed information retrieval. *Information Retrieval Journal*, 18(6), 527-549.

[12] Baeza-Yates, R., Gionis, A., Junqueira, F., Murdock, V., Plachouras, V., & Silvestri, F. (2008). Design trade-offs for search engine caching. *ACM Transactions on the Web*, 2(4), 1-28.
[13] Lin, S., Zeinalipour-Yazti, D., Kalogeraki, V., Gunopulos, D., & Najjar, W. A. (2006). Efficient indexing data structures for flash-based sensor devices. *ACM Transactions on Storage*, 2(4), 468-503.
[14] Ganesan, P., Garcia-Molina, H., & Widom, J. (2005). Exploiting hierarchical domain structure to compute similarity. *ACM Transactions on Information Systems*, 21(1), 64-93.
[15] Datta, S., Patel, R., & Kumar, V. (2021). Quorum-based replication and reconstruction codes for distributed storage. *IEEE Transactions on Dependable and Secure Computing*, 18(3), 1245-1260.
[16] Chikhaoui, B., Wang, S., & Pigot, H. (2021). Data placement optimization in distributed storage systems: A survey. *ACM Computing Surveys*, 53(6), 1-35.
[17] Ha, K., Park, J., & Lee, S. (2021). Machine learning approaches for hot data identification in tiered storage systems. *IEEE Access*, 9, 45678-45692.
[18] Manchana, P. (2020). Optimizing batch workloads in Elasticsearch clusters. *International Journal of Database Management Systems*, 12(2), 23-38.
[19] Park, S., Kim, H., & Choi, Y. (2019). Design principles for cold storage in large-scale archival systems. *ACM Transactions on Storage*, 15(3), 1-26.
[20] Li, X., Zhang, Y., & Chen, W. (2018). Resource utilization analysis in shard replication for distributed databases. *Journal of Parallel and Distributed Computing*, 121, 89-103.
[21] di Vimercati, S. D. C., Foresti, S., Jajodia, S., Paraboschi, S., & Samarati, P. (2018). Enhancing data confidentiality through strategic data swapping. *IEEE Transactions on Dependable and Secure Computing*, 15(4), 667-681.
[22] Niu, Q., Dinan, J., Lu, Q., & Sadayappan, P. (2018). A comprehensive survey of hybrid storage systems: Architectures and optimization techniques. *ACM Computing Surveys*, 51(3), 1-33.
[23] Sri, K., Reddy, M., & Kumar, P. (2017). Data mining and analytics using Elasticsearch: A practical approach. *International Journal of Advanced Computer Science and Applications*, 8(5), 234-241.
[24] Duckham, M., Nittel, S., & Worboys, M. (2011). Decentralized spatial computing: Foundations of geosensor networks. *Spatial Cognition & Computation*, 5(2-3), 171-188.
[25] Wei, L., Zhao, H., & Liu, Q. (2020). Mathematical modeling for optimal shard number determination in Elasticsearch. *Performance Evaluation*, 139, 102-118.
[26] Aldailamy, A., Abdallah, M., & Fung, B. C. M. (2018). Solr vs Terrier: A comparative performance analysis of open-source search platforms. *Information Processing & Management*, 54(6), 1158-1177.
[27] Berglund, E. (2014). Shard selection strategies in Elasticsearch distributed search. *Master's Thesis, KTH Royal Institute of Technology*.
[28] Junqueira, F., Bhagwan, R., Hevia, A., Marzullo, K., & Voelker, G. M. (2012). Surviving slashdot: Reactive index replication for peer-to-peer search. *IEEE Transactions on Parallel and Distributed Systems*, 23(8), 1481-1492.
[29] Kulkarni, A., & Callan, J. (2010). Document allocation policies for selective searching of distributed indexes. *ACM CIKM Conference*, 449-458.
[30] Qian, G. (2008). Adaptive indexing for content-based search in P2P systems. *Data & Knowledge Engineering*, 67(3), 381-398.