Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 167 (2020) 1626–1635

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Computational Intelligence and Data Science (ICCIDS 2019)

Performance Analysis of Distributed Processing System using Shard Selection Techniques on Elasticsearch

Praveen M Dhulavvagol [a,*], Vijayakumar H Bhajantri [a], S G Totad [a]

[a] *School of Computer Science and Engineering, KLE Technological University, Hubballi, 580031, India*

Abstract

In distributed systems multiple computers interact with each other over a network to perform data processing operations parallel over a network. Distributed systems can be setup depending on the application and its data with different combination of replications and then the partitioning of the data can be performed using sharding technique to enhance the scalability, consistency and fault-tolerance issues. The gaps identified in existing sharding techniques are, it is difficult to determine which particular shard or partition need to be searched to retrieve the relevant document matching to the user query. And also to rank the documents which has highest relevance to the user query. The main goal is to develop a mechanism to handle large scale data processing and searching operations using efficient sharding technique. The proposed paper aims to enhance the performance of distributed processing systems by applying effective shard partitioning and efficient shard selection techniques and perform the comparative study analysis of shard selection techniques considering precision, MAP and cost measures. The results interpret that sharding technique provides efficient mechanism to handle large scale data. Effectiveness and efficiency of shard selection techniques interpret that Rank-S algorithm performance better compared to CORI and Redde algorithms by reducing the overall cost by 28%. Sharding technique is an efficient mechanism to process big data, and provides better scalability and fault-tolerance.

Keywords: CORI; Redde; Highsim; Sharding; Precision;

1. Introduction

A distributed system architecture mainly consists of multiple data nodes which are interconnected by a network across different locations and work in the collaboration parallel, to perform data processing operations.

Corresponding Author: Praveen M Dhulavvagol; Tel: +91-9964266154
E-mail address: praveen.md@kletech.ac.in

10.1016/j.procs.2020.03.373

In centralized data processing only one computer is responsible to manage the data processing operations. This approach has the limitations like delay, error and scalability issues. The benefit of using distributed systems compared to centralized, distributed data processing provides more security, faster problem solving, and collaborative processing.
Distributed systems should be designed such that it should provide the solution to key issues:

- Where to place the data
- How to divide relations to sub-relations (Fragments)
- How to avoid replication of the data.

Many of the researchers have implemented data fragmentation technique to overcome the scalability and consistency problems in distributed systems. In data fragmentation or data partition technique the data is partitioned into different fragments to improve the search efficiency. Data partition techniques are of two type's horizontal and vertical data partitioning. In vertical data partitioning the data is partitioned column wise and each partitions are maintained separately, these partitions are called as shards. In horizontal partitioning the data is partitioned row wise, this enhances the scalability and performance of the system. Each partition or shards work in collaboration to process the data or user request without any loss of data. The data partitioning techniques are not efficient to process large size data due to the limitations like high communication overhead, scalability, consistency overhead problems and memory underutilization, To overcome the limitations of data partitioning strategies sharding technique was proposed. Sharding is a database partitioning technique that divides a data row wise and stores this data into multiple nodes which will work in collaboration parallel to achieve the required goal and enhances the performance [1].

A data store hosted by single centralized storage server may not perform efficiently when huge volume of data is processed, this may lead to following issues such as low storage space, low computing power and scalability issues. Scaling vertically by adding more disk capacity, processing power, memory, and network connections will not overcome the issues, instead it postpones the effects of these limitations. Considering the above facts vertical scaling is not the efficient solution to enhance the performance of distributed processing system. Instead partition the data store horizontally into different number of shards to enhance the scalability, consistency, and performance of data store [2]. Sharding is a technique which is mainly used to increase the performance of distributed databases by creating n number of shards or partitions. A shard or partition is portion of a database distributed across different nodes. Sharding is a key approach to scale distributed processing systems, maintaining the privacy and security features [3].
The advantages of using sharding technique are:

- Increased search performance and smaller index size: We can have N number of partitions/shards depending on the data as data is distributed in multiple shards each shard will have smaller index size and this result in faster processing and enhances the document search performances.
- Reliability: The system will be more reliable from the failures and this is one of the important goal of sharding. Multiple nodes provide high availability and data consistency is maintained and there will be minimum downtime.
- Harmonizing: sharding provides horizontal scalability by partitioning data into multiple nodes so the load is equally distributed on all the shards.
- Speed: in a distributed systems a search index can be partitioned into N different shards and load each index on a separate server. For any given query we will get 1/Nth of results. To obtain the complete result set a distributed system uses aggregator which will collect and collaborate all the results from different nodes. It also distributes the query to all the nodes this operation is known as mapreduce so the distributed system performs both sharding and mapreduce operations.

1.1 Understanding Sharding in ElasticSearch

ElasticSearch is an open source platform mainly used as a search engine in the information retrieval systems. Provides a platform that supports for wide variety of use cases, which provides great flexibility for data processing and replication operations. Elasticsearch uses sharding technique and is highly scalable because of its distributed architecture.

Elasticsearch engine provides distributed selective search operation and is based on the following hypothesis:

- A document collection can be partitioned such that the majority of the relevant documents for a query are stored in a few shards without any prior knowledge of the query-set.
- When a data is partitioned across different shards, then these shards can be ranked based on their relevance to the query.
- The minimum number of top shards in the ranking that need to be searched for the query can be estimated.

Basically there are two types of shards primary and replica shards. When data is stored in the primary shards, the same copy of data will get stored in replica shards this is useful whenever the primary shards fails or crashes then the user requests are rerouted to the replica shards so fault tolerance and data loss is very minimal with this approach.

In summary, Major contributions of the proposed work are:

- Configuration of the Elasticsearch engine to perform sharding operation
- Setup a sharded cluster and configure the shards depending on the size of data, the number of shards needs to be configured.
- Preparation of the dataset required to test it on sharded cluster i.e. data may be structured/semi structured data, for text based data TREC and GOV2 dataset is considered.
- Implementation of shard selection algorithm and test the algorithm on the data collection.
- Apply shard ranking algorithm on the shards to determine which all shards need to be referred for the given query.
- Perform comparative analysis of shard selection algorithms

The section 2 provides details of related work with different shard selection algorithms. Section 3 describes the proposed methodology with system architecture, data set and shard selection process. Section 4 deals with the implementation details of the proposed work with process flow diagrams and algorithms. Section 5 presents the results and discussion along with the Performance Analysis of different shard selection algorithms. Section 6 describes the conclusion and the future enhancement of the proposed work.

2. Related Work

Distributed data processing is being mainly used in many of the search engine system to efficiently manage the data and user queries. Many researchers have proposed multiple solutions to enhance the performance of the search engines, one of the efficient method proposed by author in [1] is data partitioning technique called sharding. Sharding technique partitions the large database into smaller indexes called as shards. Each shards acts as node in storing and managing the data. This technique provides better solution for distributed systems and blockchain applications by enhancing the performance of the distributed systems. The author proposes resource selection algorithm such as Rank-S or Taily to build the index rank and to analyse which shards and how many number of shards need to be referred to retrieve the relevant documents for the given query such that efficient utilization of the resources is done which intern will enhance the performance of distributed processing system.

Shard selections techniques are basically used to optimize the search engines. The main idea is to process the user query to only those shards were relevant document are retrieved ignoring the other shards which doesn't contain relevant data. Four different shard selection algorithms have been implemented in [2] and a comparative study is being carried out using two different data sets. Precision and recall measures are used to measure the accuracy of system. The results indicate that out of four, three algorithms have resulted in better performance [3].

In this paper [4] he author discuss on shard selections algorithm CORI, Redde and Highsim considering TREC dataset which is usually used in the information retrieval system. Shard selection is a technique which is used to forward the user queries to the broker agent or the master node. This techniques searches the document matching to the user query only in those shards which have relevant document matching to the query. If the documents are

randomly distributed to the shards then the shard selection algorithms may not give efficient results and this is one of the standard approach which is used in Elasticsearch engine. The lexicon algorithms used by author are CORI and Highsim, and surrogate algorithms are ReDDe, sushi, sampling-based hierarchical relevance estimation. The dataset used for experiment purpose are TREC-database and twitter dataset [5]. Sharding by Hash partitioning technique partitions the data across different shards, this technique is used to achieve evenly sharded database cluster and is used as one of the scalability pattern design. Applications like social networks, e-commerce, generate lots of data traffic. The database need to scale up to handle this huge volume of data. Hash based partitioning technique provides better solutions for handling a huge volume of data [10].

Selective search architecture uses resource selection algorithms such as Rank-S to rank the shards and also to estimate how many number of shards need to be consulted to retrieve the relevant data for the given query. The author discuss a new approach which will predict how many number of shards need to be consulted to increase the recall. Earlier selective search is treated as single stage retrieval method and it majorly focuses on optimizing the precision metric [9]. The author implements a new query-specific shard ranking method which will satisfy the early-precision or high recall retrieval goals [11].Taily, a novel shard selection algorithm showed better efficiency and effectiveness Compared to the popular vocabulary-based method CORI. Compared to Rank-S Taily achieved better effectiveness utilizing fewer resources.

3. Proposed Methodology

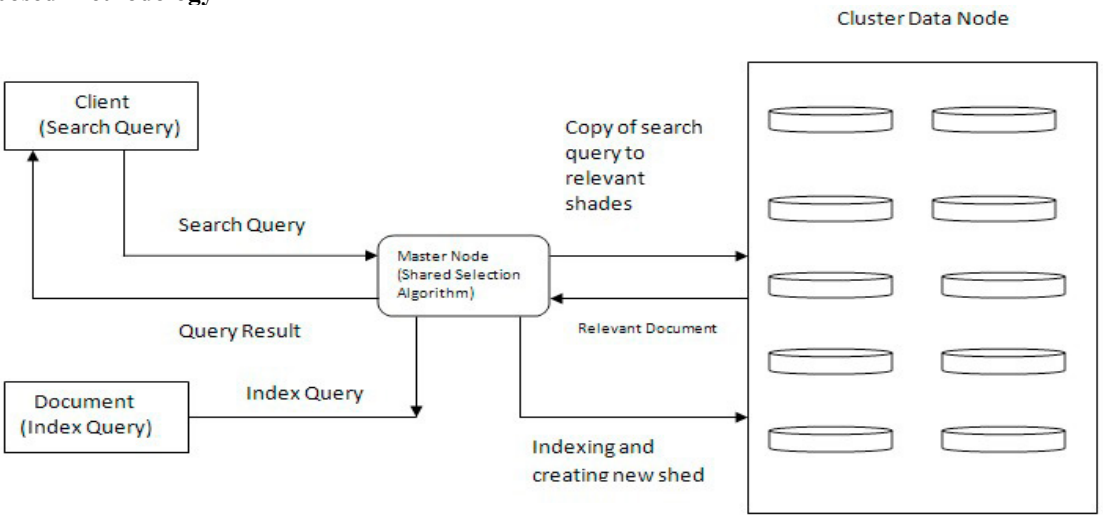

Fig.1. System architecture

In a distributed processing system all the nodes work in collaboration to process the data. The main goal is to enhance the scalability, consistency and performance of the system, so to achieve optimal performance it's important to divide the data such a way that it can handle any type of queries which the application demands. Partitioning of data in a distributed system provides a better solution compared to all other existing approaches. One of the best partitioning methods used in this case is sharding. In few cases this sharding approach may not be feasible solution for some of the queries. To handle these situations, three different sharding strategies are available such as Lookup, Range and Hash. Most common sharding systems implement one of these sharding strategies depending on the business requirement. The main goal of any distributed search engines is to fulfil the user information need by retrieving the relevant document for the given query as fast as possible [29]. The Fig. 1 shows that it's difficult to determine if a document is relevant to a query is fundamentally hard problem. The relevance is calculated based on the similarity score computed by the query and document, different methods have been already proposed such as Boolean model, vectorspace model and language models.

3.1 Shard Index Construction

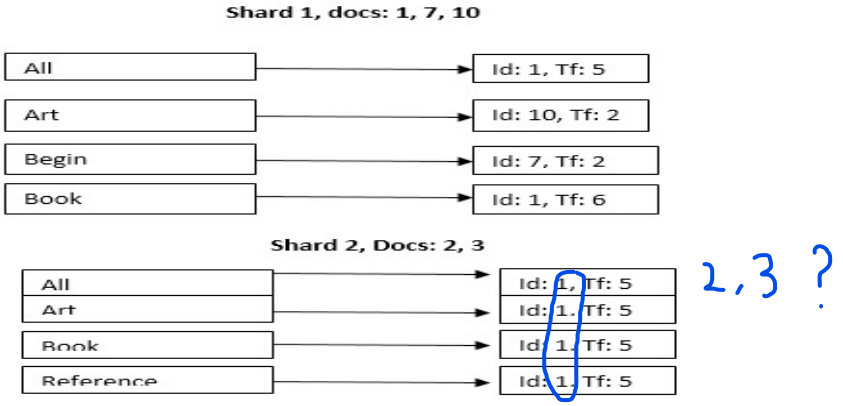

Fig .2. Dictionary and Posting List

The Fig. 2 shows the dictionary and posting list which is used for documents index construction. In the dictionary part query keywords are considered and the posting list contains documents id and term frequencies. As the number of documents increases the size of the posting list will increase and it becomes a complex and time consuming process to perform query-document search operation which intern will result in low throughput and less relevance. The optimal solution is to divide the index into smaller partitions. This process is called sharding. In the above figure two shards are created in shard 1 document id 1, 7 & 10 are stored and in shard 2 document id 2 and 3 are stored, The shards may be distributed to different nodes which enables multiple computers to collaborate in the search-process.

3.2 Document and Query Similarity Score Computation

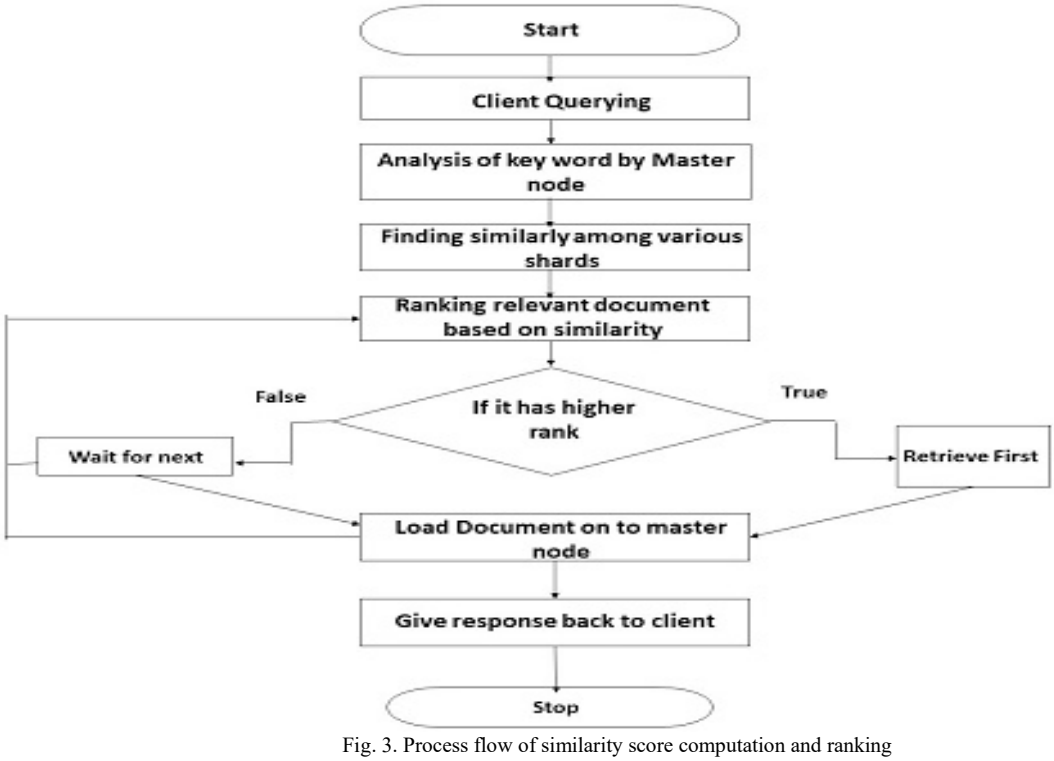

Fig. 3. Process flow of similarity score computation and ranking

The Fig. 3 shows the similarity computation and ranking process. Tf-Idf scoring model is considered to determine the similarity score between query and document, in this model the document with the highest term frequency score is considered as more relevant document to the given query. Considering only term frequency may not find relevance with respect to the user query, so term frequency are combined with inverse document frequency to compute the similarity score. Idf score is computed using equation 1.

$$Idft = \log \times N/ Dft \quad (1)$$

The term frequency (Tf) and Inverse document frequency (IDF) together determine the similarity score of the document which is computed as

$$Score (d, q) = \Sigma T ft,d \times Idft \quad (2)$$

When a client sends a query, the master node analyses the keyword and finds the similarity scores among the documents. Then documents are given ranks based on similarity score. The highest ranked documents are retrieved first.

3.3 Shard Selection Process

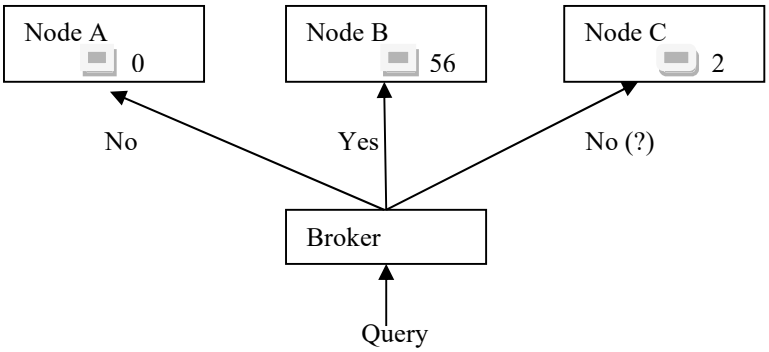

Fig.4. Shard selection process

The Fig 4 illustrates of process of shard selection process, user send the query to the broker node which will intern re-route the query request to all the connected data nodes i.e. node A, B, C, All the data nodes may not contain the relevant document matching to the user query. So in such cases only few nodes will be contributing to fetch the top most relevant documents. The most commonly used shard selection approach is to have one shard in each data node. Once the relevant documents are fetched from the data node. The broker node will integrate all the documents and returns back to the user. In the fig 1 most of the relevant documents are with node B and no relevant documents with node A and very few relevant documents with node C. the broker node should have ignored searching relevant documents from node A and C, there might be chances that node C might contain documents which are more relevant to the query compared to the documents from node B, so selection of shard plays an important role in determine relevant results for the given query and also it saves the resources and enhance the document retrieval time.

3.4 Dataset

Three datasets were used for experiment purpose, the details of these data collections are as follows: The Shakespeare's play dataset is in JSON format and contains 10000 documents under four attributes. This dataset comprised of all of Shakespeare's plays. It includes the following: The first column is the Data-Line, it just keeps track of all the rows there are. The second column is the play that the lines are from. The third column is the actual line being spoken at any given time. The fourth column is the Act-Scene-Line from which any given line is from. The fifth column is the player who is saying any given line. The sixth column is the line being spoken. And second dataset considered is 20- news group dataset which consists of approximately 20,000 documents, partitioned (nearly) evenly across 20 different newsgroups. The third dataset used is GOV2 data collection.

4. Implementation

4.1 Querying the index and Similarity Score Computation

First, we give a query (e.g., Did lately meet in the intestine shock) the search for the result takes place according to Boolean model to compute the similarity score and the document having highest score will be retrieved.
The query result retrieved will be evaluated with Tf-Idf similarity scoring function and we get the score for each index is based on the score from Tf-Idf score.

Algorithm 1: Querying the Index

```
GET /shakespeare/_rank_eval
{
    "requests":[
    {
        "id": "JFK query",
"request": { "query": { "match": { "text_entry"
            :"Did lately meet in the intestine shock"}}},
        "ratings": [
            ]
    }],
    "metric": {
        "precision": {
            "k" : 10,
            "relevant_rating_threshold": 1,
             "ignore_unlabeled": false
        }
    }
}
```

Algorithm 2: Similarity Score Computation

```
{"metric_score" : 0.1,
 "details" : {
     "JFK query" : {
         "metric_score" : 0.1,
         "unrated_docs" : [
             {
                 "_index" : "shakespeare",
                 "_id" : "2264"
             },
            {
            "_index" : "shakespeare",
            "_id" : "2925"
     }
```

4.2 Precision @ K

For any information retrieval system precision is an important metric compared to Recall, precision provides us the whole set of relevant documents which are relevant to the query and recall is to know total how many number of relevant documents have been retrieved. Precision at k documents is a very useful metric (P@k) to determine the relevance of the documents matching to the user query.

Algorithm 3: Precision@ K

```
"metric_details": {
   "precision" : {
       "relevant_docs_retrieved": 1,
            "docs_retrieved": 10
```

5. Results and Discussions

Shakespeare data collection is considered for experiment purpose which is in Json file format. And we have performed the normal sharding operation considering precision at rank 10, 20, and 30. I.e. for example, at Pk@5 means the top 5 documents will be searched for the query results. The table 1 shows the precision score at rank 10, 20, and 30 in this case we have not applied any specific shard selection algorithm, instead we have created the shards and indexed the documents in each of the shards and performed the search operation.

Table 1. Precision @ K value

Query	Pk@10	Pk@20	Pk@30
"Did lately meet in the Intestine shock"	33.3307	30.6089	29.0039
"King Henry"	5.920	5.004	4.970
"The hostilepaces"	33.681	31.719	29.786

The above Table 1 shows the average precision score value determined for the given query, more the precision score means more number of relevant documents are retrieved for the given query. The table 2 interprets the CPU time, similarity score obtained for the query, with respect to different number of shards.

Table 2. Similarity score vs. time to retrieve relevant document

No of shards	CPU Time (ms)	Score
1	42	6.5059
3	56	5.4694
6	80	4.8148

The Figures (Fig.5a, 5b, 5c) represents graphs of time taken by different set of shards for the tested queries Fig 7a represents the time taken by the query 1 (playname : "Henry IV") with only 1 shard the time taken is 6msec and with 5 shards , time taken is 4msec. Similarly, time taken is decreasing as the number of shards is increasing. Fig 7b represents the time taken by the query 2 (playname : "Henry IV", playline : "Shall daub her lips with her own children's blood") with only 1 shard the time taken is 12msec and with 5 shards , time taken is 10msec. Similarly, time taken is decreasing as the number of shards is increasing. Fig 7c represents the time taken by the query 3 (player "westmorland", playline : "Shall daub her lips with her own children's blood") with only 1 shard the time taken is 11msec and with 5 shards , time taken is 13msec. Similarly, time taken is increasing as the number of shards is increasing.

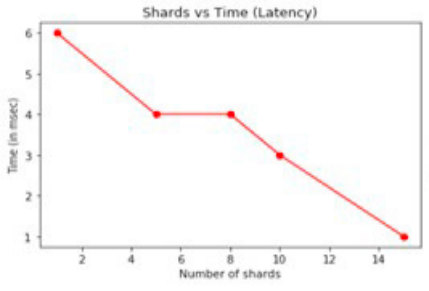

Fig.5a. Latency graph for query 1

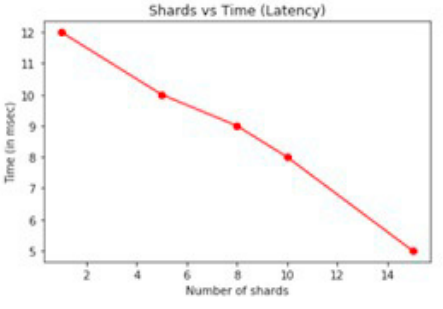

Fig.5b. Latency graph for query 2

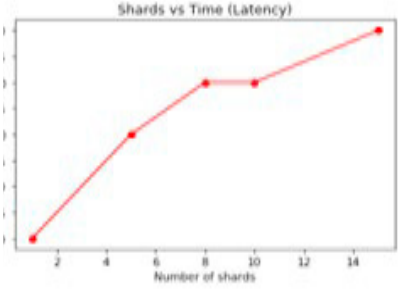

Fig.5c. Latency graph for query 3

The Figures (Fig.6a, 6b, 6c) represents graphs of document maximum scores on different set of shards for the tested queries. Fig 6a represents the document scores for query 1 (playname : "Henry IV). With only 1 shard the maximum score is 3.37 and with 5 shards score is 4.15. Similarly, score is increasing as the number of shards is increasing. The number of documents retrieved is 132. Fig 6b represents the document scores for query 2 (playname "Henry IV" , playline : "Shall daub her lips with her own children's blood"). With only 1 shard the maximum score is 0.95 and with 5 shards score is 1.07. Similarly, score is increasing as the number of shards is increasing. The number of documents retrieved is 96. Fig 6c represents the document scores for query 3 (player : "westmorland" , playline : "Shall daub her lips with her own children's blood"). With only 1 shard the maximum score is 5.0 and with 5 shards score is 5.37. Similarly, score is increasing as the number of shards is increasing. The number of documents retrieved is 32.

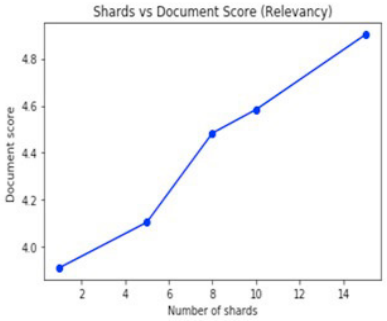

Fig.6a. Releveance graph for query 1

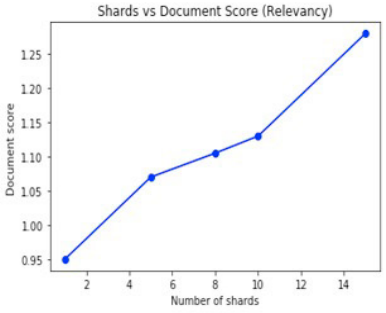

Fig.6b. Releveance graph for query 2

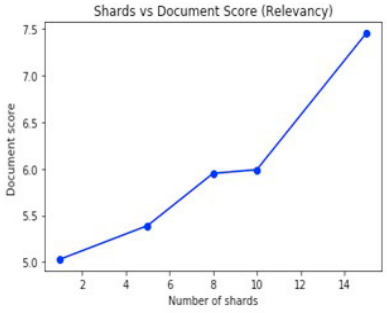

Fig.6c. Releveance graph for query 3

To perform the comparative study analysis of Redde, Rank-s and CORI shard selection algorithms, the performance metrics considered is precision for rank 10, rank 30 i.e. (P@{10, 30}). Normalized discounted cumulative gain at rank 10 (ndcg@10) and MAP score is used to estimate the average search accuracy and also cost measure is considered to check the resource and time utilization. As compared to vocabulary based algorithm CORI, sample based algorithm Rank-S performance significantly better as compared to Redde, the other state-of-art shard selection algorithm. We will measure the effectiveness of the Redde, CORI and Rank-S algorithms considering precision at 10, 30 so (P@10, P@20, P@30), MAP and ndcg at 10 (ndcg@10). Considering a single effectiveness measure then consider P@30, it provides a good results as compared to P@10 or 20 for all shard selection algorithms. The table 3 shows the comparative study of effectiveness and efficiency of Redde, Rank-S and CORI algorithms (i.e. search accuracy and cost) for GOV2 dataset. Effectiveness and efficiency are the important characteristics for any shard selection algorithms. The exhaustive search performance at P@10=0.58 and for P@0 = 0.57 and Mean Average Precision (MAP) score =0.33 and ndcg@10 =0.49 and the overall cost =3.62

Table 3. Performance analysis of shard selection algorithms

Shard Selection Methods	P@10	P@30	MAP	ndcg@10	Cost
Redde (T=3)	0.56	0.45	0.30	0.47	0.80
Rank-S (B=10)	0.57	0.51	0.28	0.47	0.60
CORI (n=3)	0.56	0.47	0.35	0.48	0.65

The above table 3 interprets the result for GOV2 data collection , for Redde the parameter considered is fixe shard rank cut-off (T), and for CORI the shard cut-off value n=3, Higher the values sometimes produce better results so we can set Rank-S parameter B value as 10 or more. For the table we can infer that the Rank-S algorithm provides the minimum cost value for the given GOV2 data collection. All the three algorithms were able to provide comparable values, but the Rank-S algorithm reduces the search cost by 28% for GOV2 data collection.

6. Conclusion

Elasticsearch platform is used to implement different sharding techniques. The main aim is to develop a mechanism to handle large scale data processing and searching operations using efficient sharding technique. Sharding technique is applied to enhance the performance of distributed processing systems by applying effective shard partitioning and efficient shard selection techniques. The comparative study analysis of shard selection techniques is performed considering precision, MAP and cost measures. The results indicates that as the number of shards increased the similarity score of query and document also increases, the most relevant document is retrieved for a given query. Finally, a comparative study analysis of different sharding techniques on distributed Elasticsearch cluster is performed and results interpret that Rank-S algorithm performance is better compared to CORI and Redde algorithms by reducing the overall cost by 28%. There is a large scope for research to investigate the potential of integrating distributed ledger technology (blockchain) with sharding technique to enhance the scalability and overall performance of distributed processing systems.

References

[1] Joarder Kamal, ManzurMurshed, RajkumarBuyya, Workload-aware incremental repartitioning of shared-nothing distributed databases for scalable OLTP applications, Future Generation Computer Systems, Volume 56, 2016, Pages 421-435, ISSN 0167-739X, https://doi.org/10.1016/j.future.2015.09.024

[2] Hafeezul Rahman Mohammad, KeyangXu, Jamie Callan, and J. Shane Culpepper. 2018. Dynamic Shard Cut-off Prediction for Selective Search. In The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '18). ACM, New York, NY, USA, 85-94. DOI: https://doi.org/10.1145/3209978.3210005

[3] Zhuyun Dai, ChenyanXiong, and Jamie Callan. 2016. Query-Biased Partitioning for Selective Search. In Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM '16). ACM, New York, NY, USA, 1119-1128. DOI: https://doi.org/10.1145/2983323.2983706

[4] Anagha Kulkarni and Jamie Callan. 2015. Selective Search: Efficient and Effective Search of Large Textual Collections. ACM Trans. Inf. Syst. 33, 4, Article 17 (April 2015), 33 pages. DOI=http://dx.doi.org/10.1145/2738035.

[5] Anagha Kulkarni, Almer S. Tigelaar, DjoerdHiemstra, and Jamie Callan. 2012. Shard ranking and cutoff estimation for topically partitioned collections. In Proceedings of the 21st ACM international conference on Information and knowledge management (CIKM '12). ACM, New York, NY, USA, 555-564. DOI: https://doi.org/10.1145/2396761.2396833.

[6] Dhulavvagol P.M., Totad S.G., Sourabh S. (2019) Performance Analysis of Job Scheduling Algorithms on Hadoop Multi-cluster Environment. In: Sridhar V., Padma M., Rao K. (eds) Emerging Research in Electronics, Computer Science and Technology. Lecture Notes in Electrical Engineering, vol 545. Springer, Singapore

[7] Feng, Xiaoqin& Ma, Jianfeng& Feng, Tao & Miao, Yinbin& Liu, Ximeng. (2018). Consortium Blockchain-Based SIFT: Outsourcing Encrypted Feature Extraction in the D2D Network. IEEE Access. 6. 1-1. 10.1109/ACCESS.2018.2869856.

[8] Joshi Y., Totad S.G., Geeta R.B., Prasad Reddy P.V.G.D. (2018) Mobile Agent-Based Frequent Pattern Mining for Distributed Databases. In: Bhalla S., Bhateja V., Chandavale A., Hiwale A., Satapathy S. (eds) Intelligent Computing and Information and Communication. Advances in Intelligent Systems and Computing, vol 673. Springer, Singapore

[9] Narayanan Venkateswaran, Dr SuvamoyChangder Simplified Data Partitioning in a Consistent Hashing Based Sharding Implementation. Proc. of the 2017 IEEE Region 10 Conference (TENCON), Malaysia, November 5-8, 2017.

[10] Costa, Caio H. et al. "Sharding by Hash Partitioning - A Database Scalability Pattern to Achieve Evenly Sharded Database Clusters." ICEIS (2015).

[11] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta and B. Ford, "OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding," 2018 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, 2018, pp. 583-598. doi: 10.1109/SP.2018.000-5.

[12] P. M. Dhulavvagol, A. Desai and R. Ganiger, "Vehical Tracking and Speed Estimation of Moving Vehicles for Traffic Surveillance Applications," *2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC)*, Mysore,2017,pp.373-377. doi: 10.1109/CTCEEC.2017.845504

[13] Kurt Rohloff and Richard E. Schantz. 2010. High-performance, massively scalable distributed systems using the MapReduce software framework: the SHARD triple-store. In Programming Support Innovations for Emerging Distributed Applications (PSI EtA '10). ACM, New York, NY, USA, Article 4, 5 pages. DOI=http://dx.doi.org/10.1145/1940747.1940751

[14] LoiLuu, Viswesh Narayanan, ChaodongZheng, KunalBaweja, Seth Gilbert, and PrateekSaxena. 2016. A Secure Sharding Protocol for Open Blockchains. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16). ACM, New York, NY, USA, 17-30. DOI: https://doi.org/10.1145/2976749.2978389

[15] Niranjan C Kundur1, Praveen M Dhulavvagol2, Prasad." Recommendation System Based on Content Filtering for Specific Commodity" International Journal of Latest Technology in Engineering, Management & Applied Science (IJLTEMAS)Volume V, Issue VII, July 2016 | ISSN 2278-2540

[16] Hussam Abu-Libdeh, Robbert van Renesse, and YmirVigfusson. 2013. Leveraging sharding in the design of scalable replication protocols. In Proceedings of the 4th annual Symposium on Cloud Computing (SOCC '13). ACM, New York, NY, USA, Article 12, 16 pages. DOI: https://doi.org/10.1145/2523616.2523623

[17] MuthukaruppanAnnamalai, KaushikRavichandran, Harish Srinivas, Igor Zinkovsky, Luning Pan, Tony Savor, David Nagle, and Michael Stumm. 2018. Sharding the shards: managing datastore locality at scale with Akkio. In Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation (OSDI'18). USENIX Association, Berkeley, CA, USA, 445-460.

[18] S G Totad, R B Geeta, CR Prasanna, NK Santhosh, PV Reddy." Scaling data mining algorithms to large and distributed datasets" International Journal of Database Management Systems (IJDMS), Vol.2, No.4, November 2010

[19] I. Weber et al., "On Availability for Blockchain-Based Systems," 2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS), Hong Kong, 2017, pp. 64-73. doi: 10.1109/SRDS.2017.15

[20] Y. Liu, Y. Wang and Y. Jin, "Research on the improvement of MongoDB Auto-Sharding in cloud environment," 2012 7th International Conference on Computer Science & Education (ICCSE), Melbourne, VIC, 2012, pp.851-854. doi: 10.1109/ICCSE.2012.6295203

[21] Dhulavvagol P.M., Ankita K.R., Sohan G., Ganiger R. (2018) An Enhanced Water Pipeline Monitoring System in Remote Areas Using Flow Rate and Vibration Sensors. In: Nagabhushan T., Aradhya V., Jagadeesh P., Shukla S., M.L. C. (eds) Cognitive Computing and Information Processing. CCIP 2017. Communications in Computer and Information Science, vol 801. Springer, Singapore

[22] Yashaswini Joshi, Geeta R.B, Shashikumar G. Totad, and Prasad Reddy PVGD, "Mobile Agent based Frequent Pattern Mining for Distributed Databases", (Springer sponsored) International Conference on Intelligent Computing and Communication (ICICC - 2017) on 2 to 4th August 2-4, 2017 at MAEER's MIT College of Engineering, Pune; Intelligent Computing and Information and Communication(Springer Book Chapter), 20-01-2018,pp:77-85,DOI:10.1007/978-981-10-7245-1_9.

[23] Lin J, Ryaboy D, Weil K. Full-text indexing for optimizing selection operations in large-scale data analytics. San Jose, California, New York, USA: ACM; 2011.Available from: http://www.acm.org/publications.