A Big Data Framework for Criminal Investigation using Call Detail Records

Mohamed A. Zawra
Faculty of Computers and Artificial Intelligence, Helwan University, Egypt

O.E. Emam
Faculty of Computers and Artificial Intelligence, Helwan University, Egypt

M. Elemam.Shehab
Egyptian Armed Forces, Egypt

ABSTRACT
Call Detail Records (CDRs) can be considered as big data source as it has a huge volume, variety of data and high data rate, as well. The analysis of CDRs can produce big value and offer opportunities to maximize revenue and improve the community's standard of living. However, the analysis of such data with those characteristics calls for using big data technology. Big data analytics is a rapidly growing field that has the potential to revolutionize the way we handle various aspects of our lives. One area where this technology is particularly relevant is in criminal investigations, where the analysis of call detail records (CDRs) can provide valuable insights into the activities and movements of individuals associated with criminal activity. This paper proposes a framework that leverages the massive amounts of data generated by telecommunication networks to uncover valuable insights that can help criminal investigators. Furthermore, the proposed framework is optimized to reduce the underlying computing resources needed to analyze such huge amount of data and to improve the overall performance of the proposed framework, as well.

Keywords
Call Detail Records (CDR), Big data analytics, Criminal investigation.

1. INTRODUCTION
The rapid expansion of the telecommunication network and the widespread use of mobile phones have generated a huge amount of data, including Call detail Records (CDRs). CDR is a data record that contains information about a telephone call or communication session, including details such as the time, date, duration, source number, destination number, Cell/Location ID, and any other relevant information about the call. CDRs are typically generated by telecommunication service providers or telephone companies for billing purposes, and they can also be used for charging, settlement, network analysis, troubleshooting, fraud detection, churn detection and security purposes.

In addition, CDRs can be used by law enforcement agencies to investigate criminal activities such as fraud, harassment, and other types of illegal activities. The call detail records have proven to be valuable in various sectors, including criminal investigation, as they provide insights into communication patterns, movement, and activities of individuals [1-10].

Due to the huge volume of the CDRs, the manual analysis of that data can be overwhelming for investigators. However, with the help of big data analytics tools, it is possible to identify patterns and relationships that would otherwise be impossible to discover. One of the key benefits of using big data analytics in criminal investigations is the ability to quickly identify potential criminals [4].Another benefit is the ability to track suspects and their movements [11]. Furthermore, big data analytics can also be used to identify and analyze relationships between individuals, which can provide valuable insights into criminal networks [3].Therefore the advancement of big data technology makes the analysis of CDRs necessary and enhances the effectiveness and efficiency of criminal investigations.

The rest of the paper is organized as follows. Section 2 reviews the previous work related to analyzing CDRs to generated valuable insights. Section 3 introduces the proposed framework. The experimental results are discussed in section 4. Finally, section 5 concludes the paper and sketches out the future work.

2. LITERATURE REVIEW
This section reviews the previous works that analyze CDRs and phone records to generate valuable insights in many different areas such as crime investigation, network optimization, and event detection.

A new model that uses graphs and data analysis techniques to investigate crimes using Call Detail Records has been proposed in [1]. The authors suggested using Neo4j, a graph database, to store and analyze the CDR data. The model focuses on understanding how users behave and extracting valuable information from the CDR data.

The authors in [2] discussed the importance of analyzing phone records in criminal investigations due to the widespread use of cell phones by both ordinary people and potential criminals. That study presented a graphical analytic model for the CDR database, which includes all past crimes and relationship between them, which if discovered, can help law enforcement agencies solve numerous cases. The main objective was to store all historical instances in a single centralized system for further and future analysis, rather than junk them. This helps to maintain connections and uncover trends in a graphical format.

Another study [3] proposed a new model that employs graph technologies to analyze CDRs in order to find potential criminals. Specifically, the Neo4j graph database management system is used to store the CDR in forms of graph. Then, that graph data is analyzed in an attempt to detect abnormal behavior which might help the police investigators to find links between various suspects.

Research [4] contributes to effective crime investigation by proposing a framework that enables efficient storage, retrieval, and analysis of CDR data. The proposed framework categorized the suspects to primary and secondary suspects via three-step process. In the first step, the CDRs related to the requested mobile number(s) are extracted from Hadoop distributed file system where the global CDRs are stored. Then, the extracted data is converted and stored in Hive data warehouse. Finally in the last step, the data is analyzed to

identify the most suspicious individual, and the result is provided as the output.

Authors in study [5] explore the application of big data analytics and machine learning algorithms for crime prediction. The study uses a large dataset of crime records from Chicago and shows the importance of big data analytics in understanding and combating crime in urban areas. The study builds a big data analytics model for crime prediction and evaluates the performance of different machine learning algorithms including SVM and K-NN, Random Forest, and MLP. The findings show that SVM algorithm is the best algorithm for achieving high accuracy in crime prediction within a reasonable time frame.

A new method that utilizes the CDRs to extract the abnormal communications of a given mobile network by using PageRank algorithm is presented in [6]. The proposed method takes the number of calls and the total call duration as inputs and uses a weighted version of the PageRank algorithm to analyze the impact of the communication flows on the whole network. The analysis results in detecting the abnormal communication patterns. Furthermore, combining the resulting patterns with the real events enables monitoring the traffic in real time.

A hybrid model for detecting anomalies in mobile phone networks is proposed in [7]. The model combines GARCH, K-means, and neural networks to identify anomalies, determine their causes, and gain insights into user behavior. The study shows that the hybrid model achieves lower false positive rates and higher accuracy compared to previous studies. The practical implications of this approach for mobile phone operators include real-time detection of security threats and technical issues, improving network performance, and enhancing user experience.

In order to analyze the human movement patterns from CDR data, authors in study [8] proposed a data fusion approach to predict hidden visits and differentiate between trips and displacements. The proposed method involves three main steps which are (i) Localization, (ii) Movement-state-identification, and (iii) Hidden visit inference. The experimental results of this study show that the propose method outperforms the other methods employed by the previous studies.

Research [9] is another study that focuses on analyzing human movement patterns using Call Data Records. The research objective of that study was to reduce the error in localizing the user by considering the load sharing effects of the CDR records and transmit power of the cell towers. Specifically, the study addressed two main issues: the load sharing effect and localization errors in CDRs. The research uses a large dataset of SIM cards from a mobile operator in Sri Lanka and proposes a method for preprocessing the data and reducing localization errors. User profiling techniques are used, and a load sharing record identification interface is introduced. The study demonstrates the potential of using CDRs and mobile data for transportation planning and analysis.

The study [10] focuses on using Call Detail Records from mobile networks to analyze network traffic patterns and develop a machine learning model for classification. The research aims to optimize network resources and enhance service quality by understanding spatial and temporal dependencies of network traffic. To do that, the authors perform spatiotemporal analysis of CDR data at first. Then, they employed a clustering algorithm to categorize the mobile traffic patterns. Finally, they exploit the clustering results to train a neural network in order to classify the network traffic. The findings demonstrate that the actionable insights gained from CDR data analysis can be used for network optimization, resource allocation, and improving network deployment and operation.

An efficient and user-friendly system (called swift) for analyzing Call Detail Records (CDRs) and Tower Dump (TD) data is presented in [11] to aid in crime investigations. The proposed system employs data mining techniques such as clustering and frequent pattern analysis to quickly identify local suspects and analyze their movements. It also integrates prediction techniques and mobile GIS for tracking suspect locations, providing comprehensive analysis of relevant patterns in CDR and TD data to aid in criminal investigations.

Study [12] focuses on utilizing big data technologies, specifically Hadoop, for analyzing Call Detail Records (CDRs) in the telecom industry. The main goal of the study was to extract valuable insights from large volumes of CDR data for revenue maximization, network efficiency improvement, and customer service enhancement. The authors of that study discussed the challenges in analyzing CDRs, the benefits of using big data technologies, and presented a system for storing CDR data in Hadoop, performing ETL operations, clustering the data, and analyzing the impact of pricing and bundling schemes on customer behavior.

In another research [13], authors focused on the use of big data technologies for analyzing Call Detail Record (CDR) data in the telecom industry. CDR data presents big data challenges due to its volume, velocity, and variety. Therefore, the research discussed the challenges in loading, processing, and optimizing time and resources for analyzing CDRs. It compared two main approaches, parallel DBMS and MapReduce, for big data analysis of CDRs, presented examples of CDR analysis using these approaches and highlighted their advantages and performance in different scenarios.

A real-time framework for analyzing calling patterns and behaviors of mobile phone users using only Call Detail Records (CDR) data is presented in [14]. The framework utilizes evolving fuzzy systems to cluster caller behavior and detect outliers. The dataset used in the research contains 9,834 calls, and each CDR record includes caller and receiver identifiers, call time, duration, and tower location. The proposed solution adapts to evolving data by updating clusters recursively, providing a means to extract calling patterns and behaviors solely from CDRs.

Finally paper [15] introduces a framework for detecting social events based on the location of users using mobile phone data. The framework aims to detect dangerous situations and predict the location of events such as rock concerts, sports events, protests, emergencies, and crises. The proposed method involves probabilistic location inference and clustering algorithms. The paper discusses the challenges of analyzing sparse and noisy mobile phone data and proposes a Bayesian approach for location inference. The method is evaluated using a dataset from the Dutch Royal Wedding. The findings demonstrate the high accuracy of the proposed method in detecting social events and highlight its potential applications in urban planning, event management, and public safety. The results emphasize the importance of considering uncertainty in location data, as well.

The research papers reviewed in this section show how the analysis of phone records and related data can be very helpful for many applications, such as crime investigation, network optimization, and event detection. The papers proposed many different methodologies, models, and algorithms to analyze CDRs and overcome the related challenges. They also provide valuable insights and directions for future research in areas such as transportation planning, anomaly detection, crime prediction, and network traffic optimization.

3. THE PROPOSED ARCHITECTURE

This section presents the proposed framework that includes all the components that are necessary to ingest daily CDRs data, query the records and do the necessary statistics and correlations between the captured information. The main objective of the proposed framework is to reduce latency of the analysis and increase the throughput during data ingestion and data querying. The overall architecture of the proposed framework is shown in figure 1. The framework consists of four main components: (i) Data ingestion, (ii) Data analysis, (iii) Data visualization part, and (iv) Monitoring and optimizing ElasticSearch. These four components are discussed in the following subsections.

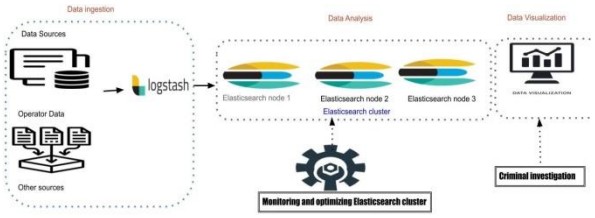

Figure 1: The Overall Architecture of the proposed Framework

3.1 Data ingestion

Data ingestion component is used to ingest CDRs data from different data sources. Then it transforms and loads the data into the ElasticSearch cluster. Data sources involve the operator data and data from other sources. The operator data may be stored in relational database, files, and message queues. Logstash [16] is used as ETL in the proposed framework. It supports the ingestion of data from multiple sources like RDBMs, files, and message queues and can be Scaled-out by providing multiple instances to increase the ingestion rates when necessary with relatively small server resources footprints.

3.2 Data analysis

The main objective of this component is to help the criminal investigators to conduct the required analysis in order to find the potential suspects. The analysis is tailored to the queries posed by the investigators. The ElasticSearch is used as a database to fulfill the performance and scalability requirements for the proposed framework [17]. ElasticSearch is a document database that supports indexing data and supports full text search in milliseconds-to-seconds response time. It helps the investigators to formulate the questions they want to ask and then it answers that questions and draws conclusions. Finally, the analysis results are visualized through the visualization component.

3.3 Data visualization

This component visualizes the results of criminal analysis. For example, figure 2 views the relationship graph or social network for a specific subscriber.

Figure 2: Social Network for Subscriber

As another example, figure 3 illustrates the results of the analysis related to tracking the movement profile or the heat map for a subscriber.

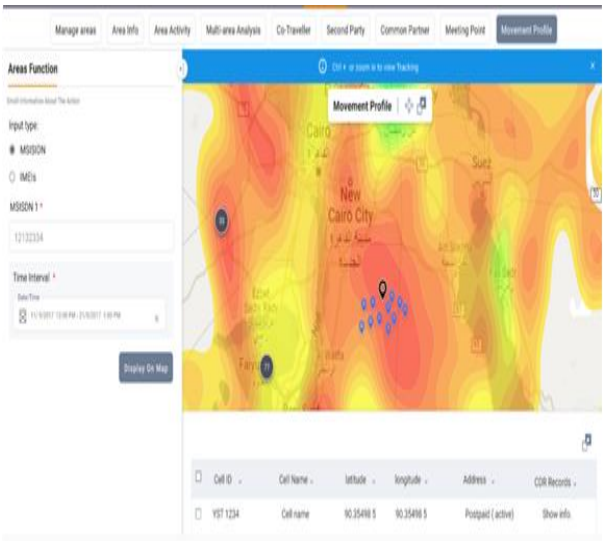

Figure 3: Movement Profile for Subscriber

3.4 Monitoring and optimizing Elastic Search

This component monitors the indexing rate, search rate, indexing latency, searching latency, shard size over time, the utilization of the resources such as the CPU consumption, and thread count. Many different performance experiments have been conducted through this component to optimize the underlying computing resources and improve the overall performance. All the details related to the conducted experiments and the results are presented in the following section.

4. EXPERIMENTS AND RESULTS

The experiments have been conducted on a cluster consists of two servers. The first server has CPU with 48 cores, 64 GB of RAM, and 2 HDDs giving a total 3 TB of disk storage. The second server has CPU with 48 cores, 64 GB of RAM, and an SSD of 400 GB. The dataset used for the most of conducted experiments are collected from Egyptian telecom operators and contains about 500,000 records of CDR data.

The experiments divided into two types: (i) indexing throughput optimization, (ii) evaluating the effect of the shard size, the type of search queries, number of concurrent requests, bulk size on the analysis throughput. The following subsections present the experimental results of those two types of experiments.

Indexing Throughput Optimization

In the first kind of the conducted experiments, the Batch Size, number of workers, and the type of the input data source (Database source vs File System based source) are factors used to test the optimization of the indexing throughput. The main goal of this type of experiments is:

- Figuring out the optimal number of workers and batch sizes that will achieve the maximum indexing throughput using a Database source
- Figuring out the optimal number of workers and batch sizes that will achieve the maximum indexing throughput using the File System besides comparing the speed of the data availability of the File System with that of the Database.

Experiment 1: Evaluating Batch size with number of workers

In this experiment, we wanted to find the appropriate batch size to use for a single worker. We have the following Logstash configurations: pipeline.workers is 1, different batch sizes (7000, 8000, 10000, 15000), queue.type: persisted and default database fetch size (jdbc_fetch_size) which is 10 rows. As shown in table 1, the indexing rates are not more than 5000 events/sec.

Table 1: Indexing rate with single worker and different batch sizes with default fetch size

Trial No.	No. of Workers	Batch Size	Queue Max	Avg Indexing Rate (docs/sec)	Avg ES CPU%	Avg LS CPU%
1	1	7000	10000	5000	1	3.5
2	1	15000	10000	4000	1	3
3	1	10000	10000	5100	1	4
4	1	8000	10000	5000	1	3.5

In an attempt to increase the indexing rate, we change the variable of jdbc_fetch_size to be 8000, 16000 and 20000 as shown in table 2. The results show that the indexing rate has increased slight a bit up to around 5300 doc/s and of average 5200 doc/sec along with slight improvements in CPU utilization.

Table 2: Indexing rate with single worker and different batch sizes with different fetch size

Trial No.	No. of Workers	Batch Size	Queue Max	Avg Indexing Rate (docs/sec)	Avg ES CPU%	Avg LS CPU%
1	1	8000	8000	10000	5200	1
2	1	8000	16000	10000	5200	1
3	1	10000	20000	10000	5300	1

We changed the number of workers to be 6, 24 and 48 as depicted in table 3. Doubling the number of workers seems to have no effect and the CPU utilization did not increase above 9%.

Table 3: Indexing rate with different number of workers and different fetch size

Trial No.	No. of Workers	Batch Size	Queue Max	Avg Indexing Rate (docs/sec)	Avg ES CPU%	Avg LS CPU%
1	24	10000	20000	10000	9000	2
2	48	10000	20000	10000	9000	2
3	6	10000	20000	10000	8500	2
4	6	10000	60000	10000	8500	2

When the Logstash Queue is used instead of reading data directly from database, the indexing rate has increased to 22000 as shown in table 4.

Table 4: Indexing rate with using Logstash queue

Trial No.	No. of Workers	Batch Size	Queue Max	Avg Indexing Rate (docs/sec)	Avg ES CPU%	Avg LS CPU%
1	6	10000	60000	1gb	22000	6
2	6	10000	60000	5gb	23000	N/A

When the number of workers is increased, the indexing rates are increased as well, as shown in table 5.

Table 5: Indexing rate when using queue and increasing number of workers

Trial No.	No. of Workers	Batch Size	Queue Max	Avg Indexing Rate (docs/sec)	Avg ES CPU%	Avg LS CPU%
3	12	10000	60000	5gb	33000	N/A
4	12	15000	60000	5gb	33000	N/A
5	24	10000	60000	5gb	35000	9

As illustrated in the above experiments, using the full-queue mechanism leads to better results when compared to using the database.

Experiment 2: The effect of the type of the input data source (file vs database)

This experiment checks the effect of using a single instance of Logstash with different chunk size when keeping the number of workers and batch size constant. The results shown in table 6 illustrate that except for chunk_size of 100, all results are almost similar regarding indexing rates and CPU utilization. Furthermore, increasing chunk size above 10000 does not have any impact given the same configuration parameters.

Table 6: Using file as a source with different chunk size

Trial No.	No. of Workers	Batch Size	Queue Max	Avg Indexing Rate (docs/sec)	Avg ES CPU%	Avg LS CPU%
1	9	3000	1048576	18500	4	15
2	9	3000	100	15000	4	15
3	9	3000	20000	18500	4	15
4	9	3000	777000	18500	4	15
5	9	3000	7000000	18500	4	15
6	9	3000	10000	18500	4	15

Another experiment has been conducted to evaluate the effect of using a single instance of Logstash, batch size of 3000, chunk size of 10000, and increasing the number of workers from 6 to 48. The findings shown in table 7 illustrate that there is no change in the indexing rate when using 48, 24, 12, and 9 workers. For 6 workers, however, the indexing rate is ever-so-slightly less and Logstash starts queuing. We decided that we will proceed with 9 workers.

Table 7: Using file as a data source with different number of workers

Trial No.	No. of Workers	Batch Size	Queue Max	Avg Indexing Rate (docs/sec)	Avg ES CPU%	Avg LS CPU%
7	48	3000	10000	18500	5	15
8	24	3000	10000	18500	5	15
9	12	3000	10000	18500	5	15
10	9	3000	10000	18500	5	15
11	6	3000	10000	18000	5	15

When running more Logstash instances side by side on the same server has a high impact on gaining a better indexing rate, as shown in table 8. Therefore, we recommend using different instances of Logstash because it uses separate Queues to do the job.

Table 8: Using Queue as a data source with different number of instances of Logstash

Trial No.	No. of Instances	Pipeline	No. of Workers	Batch Size	Queue Max	Avg Indexing Rate (docs/sec)	Avg ES CPU%	Avg LS CPU%
6	2	CDR	9	3000	10000	16000		15
		CDR	9	3000	10000	16000		15
		Total				32000	9	
7	3	CDR	9	3000	10000	13500		15
		CDR	9	3000	10000	13500		15
		CDR	9	3000	10000	13500		15
		Total				41000	13	

Evaluating the effect of the shard size, the type of search queries, Number of Concurrent Requests, Bulk Size on the Analysis Throughput

In the second type of experiments, the Shard Size, Bulk Size, Number of Concurrent Requests (clients), and the type of search queries are the factors that are used to evaluate the analysis Throughput.

4.2.1 Shard Size Factor

The indexing process and the searching process are two main ElasticSearch operations of the highest resource-demanding. Hot-Warm architecture in ElasticSearch suggests that, with the time series data that are indexed over time, if some nodes are dedicated for just searching operations that nodes are called "Warm" nodes. While, the Hot nodes receive both indexing and searching requests. In order to get better performance combined with better hardware costs, only the Hot nodes should be built using high-cost hardware and the other Warm nodes can be built using low-cost hardware. This also helps to increase the amount of the stored data since it can reside, after being indexed, on the Warm nodes.

To mimic the behaviors of the different types of nodes we decided to:

1. Test the effect of the shard size factor on the indexing rate during indexing-only load to have a basic run that we can compare different benchmarks against it.

2. Test the effect of the shard size factor on the searching rate during searching-only load and this would mimic the behavior of the Warm nodes.

(1) Evaluating Shard Size on Indexing Rate

At this test we indexed 520 million documents (193 G) of data using 8 parallel indexing clients with 5000 bulk size on a single CDR shard to test the effect of increasing the shard size on the indexing rate. No searching activity is involved at this phase. The bulk size has been chosen of medium size and is consistent throughout the experiment, so it should not be a factor in the collected results.

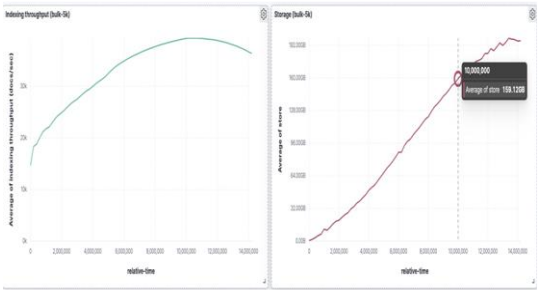

Figure 4: Shard Size vs Indexing Rate Only

(2) Evaluating Shard Size on Searching Rate

In this experiment, we indexed documents with incremental shard size of 27G, 60G, and 90G of CDR data. We issued 60 basic search operations on each size and monitored the effect of the increased shard size on the search latency and the search throughput.

As depicted on figure 5, the throughput of searching at different shard sizes up to 90 G were the same (60 ops/sec) which means that up to that size, ElasticSearch can still perform the 60 basic search operations within a sec. The search service times are of averages 40 ms, 45ms, 60 ms for the sizes 27 G, 60 G, 90 G, respectively.

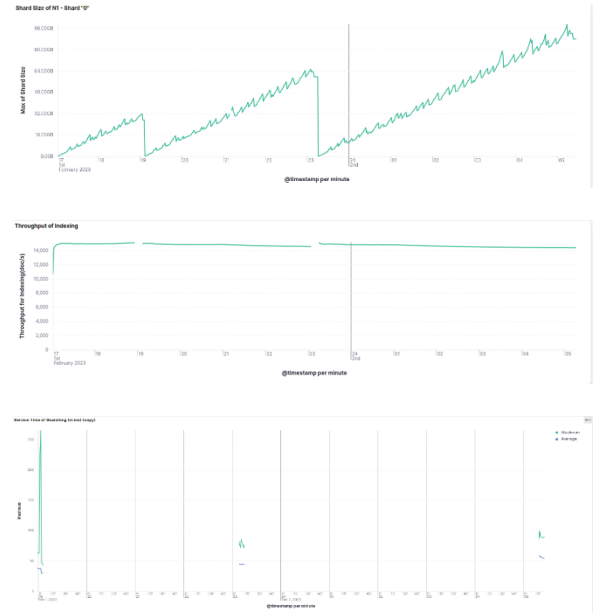

Figure 5: Shard Size vs Searching Rate

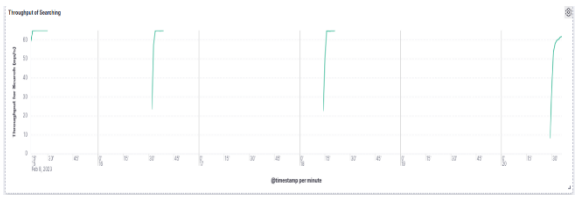

Figure 6: Throughput of Searching

4.2.2 Type of Search Query

In this experiment, we indexed incremental shard sizes of 40 G, 80G, 125G, and 170G of CDR data in one shard then issued 60 mixed search operations (match queries, aggregation queries, AND & OR operators joined queries, wild card queries, with and without time range conditions) per sec for 10 minutes (including 3 min warm-up) on each shard size and monitored the effect of increasing the shard size on the search service time and the search throughput.

As depicted in figure 6, the average of the search throughput for the 4 shard sizes is 64 ops/sec, 63 ops/sec, 62 ops/sec, and 56 ops/sec, respectively. This means that the throughput decreased when increasing the size but within an acceptable level due to the increased data size. The mixed search is showing a higher response time when compared with the results of the basic search.

4.2.3 Number of Concurrent Search Clients Factor (parallel clients)

This experiment tests how the searching throughput effected by using different numbers of parallel basic search queries (16, 32, 48, 64, 80, 96, 112, 128, 144) and shard of size 107 G on ES node that have 48 CPU threads . The results ensure that search service time increases when increasing the number of parallel requests whether the parallel requests exist in a concurrent CPU cycles for all the existing cores or not.

4.2.4 Bulk Size Factor

In this experiment, we vary the bulk size from 10k to 30k using increments of 2k and monitor the indexing rate. The results

show that the indexing has a consistent rate of around 15.5k docs/sec with all the used bulk sizes. We can conclude that working on bulks with small size finishes faster because the ES can execute more than one of them at the same time. On the other hand, working on bulks with larger size takes more time but index a huge amount of data. Therefore, both ways seem to have the same effect.

5. CONCLUSION

This paper proposes a framework for analyzing CDRs using big data tools and techniques. The framework helps the low enforcement agencies in criminal investigation. Many performance tests have been conducted to optimize the proposed framework in terms of reducing the required computing power and to improving the overall performance at the same time. Future research can explore the use of different machine learning methods to analyze CDRs in order to extract hidden patterns that can guide the criminal investigation. Another area of future research is to build real-time crime detection models by analyzing the CDRs streams in order to detect the crimes when they occur

6. REFERENCES

[1] Abuhamoud, N., &Geepalla, E. (March 2019). Analysis CDR for Crime Investigation using graph-based method (Neo4j). Retrieved from https://www.researchgate.net/publication/334587978.

[2] Kumar, M., &Hanumanthappa, M. (2016) Crime Investigation and Criminal Network Analysis Using Archive Call Detail Records. Proceedings of the 2016 IEEE Eighth International Conference on Advanced Computing (ICoAC).

[3] Abuhamoud, N., &Geepalla, E. (December 2019), "A Study of Using Big Data And Call Detail Records For Criminal Investigation in https://www.Suj.sebhau.edu.ly.

[4] Khan, E. S., Ansari, F., Dhalvelkar, H. A., &Sabiqua. (2017). Criminal Investigation Using Call Data Records (CDR) through Big Data Technology, International Conference on Nascent Technologies in the Engineering Field (ICNTE-2017).

[5] Ibrahim, S. E., &Reyad, C. A. (2023, May). A Proposed Big Data Analytics Model for Crimes Predication based on Spatial and Temporal Criminal Hotspot. Presented at CompuNet 31.

[6] Goergen, D., Mendiratta, V., State, R., & Engel, T. (2014). Analysis of large Call Data Records with Big Data. In Proceedings of the IPTComm Conference (ISBN: 1569985267).

[7] Mokhtari, A., Ghorbani, N., &Bahrak, B. (2022). Aggregated Traffic Anomaly Detection Using Time Series Forecasting on Call Detail Records. Security and Communication Networks, Volume 2022, Article ID 1182315, 9 pages. https://doi.org/10.1155/2022/1182315.

[8] Zhao, Z., Koutsopoulos, H. N., & Zhao, J. (2022). Identifying hidden visits from sparse call detail record data. Transactions in Urban Data, Science, and Technology, 1(3-4).

[9] Ayesha, B., Jeewanthi, B., Chitraranjan, C., Perera, A. S., &Kumarage, A. S. (2021). User Localization Based on Call Detail Record. arXiv preprint arXiv:2108.09157v1 [cs. LG]. Retrieved from https://arxiv.org/abs/2108.09157v1.

[10] Zhang, Z., Ahmad, A., Ali, H., & Sultan, K. (2019). Call Details Record Analysis: A Spatiotemporal Exploration toward Mobile Traffic Classification and Optimization. Information, 11(6), 192. DOI: 10.3390/info11060192. Retrieved from https://www.mdpi.com/journal/information.

[11] Singh, R., Agivale, A., Mane, M., Oza, B., &Naik, A. (2017). CDR and TD analysis using Data Mining. International Journal of Advance Research and Innovative Ideas in Education, Vol-3, Issue-6, ISSN(O)-2395-4396.

[12] Ghotekar, N. (2016). Analysis and Data Mining of Call Detail Records using Big Data Technology. International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE), Vol. 5, Issue 12.

[13] Elagib, S., Olanrewaju, R., &Hashim, A.H.A. (2015). CDR analysis using Big Data technology. In Proceedings of the International Conference on Computer, Network, and Electrical Engineering (ICCNEEE) (pp. 1-4).

[14] Iglesias, J.A., Ledezma, A., Sanchis, A., &Angelov, P. (2017). Real-Time Recognition of Calling Pattern and Behavior of Mobile Phone Users through Anomaly Detection and Dynamically Evolving Clustering. Applied Sciences, 7, 798. doi:10.3390/app7080798.

[15] Traag, V., Browet, A., Calabrese, F., &Morlot, F. (2011). Social Event Detection in Massive Mobile Phone Data Using Probabilistic Location Inference. HAL Open Science.

[16] Logstash :https://www.elastic.co/logstash (last accessed on June 2023)

[17] ElasticSearch: https://www.elastic.co/what-is/elasticsearch (last accessed on June 2023)