Handling big data security and Service through bilinear pairing and parallel computation

Ms. Shivani S. Kaktikar [1], Prof. Sarika V. Bodake [2]

[1] *M. E. Student, Computer Department, PVPIT Pune, Maharashtra, India*
[2] *Head of Department & Guide, Computer Department, PVPIT Pune, Maharashtra, India*

ABSTRACT

A new type of data, Big Data, has emerged as a result of the increased use of online services by a huge number of people. Recent years have seen an explosion in the interest in big data as a concept. Analyzing and properly processing large amounts of data is exceedingly challenging. It becomes a gigantic challenge to maintain the privacy of big data, which has an incredibly high velocity and volume which increases the possibility that mishandling of this data might lead to the theft of secret or sensitive information, which can be extremely troublesome in the case of an incident. Several techniques have been developed to counteract this impact, but they have not proven very effective. Because of this, a methodology has been developed that efficiently divides, classifies, and splits massive data query that are mapped for parallel computing on to the Mongo DB. The technique described in this research paper makes use of the idea of bilinear pairing through the application of hashing and the detection of avalanche effect for the purpose of tamper detection and forensic analysis for effective security report creation. The detailed experimentation has been performed to determine the performance metrics of the presented technique which has reached the desired outcomes.

Keyword: - *Big data, Parallel Computation, Bilinear Pairing, Avalanche effect, Data Mapping*

1. INTRODUCTION

In past few years, big data processing has had a huge influence. When it comes to large data security and management, however, additional problems emerge when the analysis is conducted from a security standpoint. The increasing interconnectivity of computer networks is accountable for the conditions that allow us to send massive volumes of data throughout the world. In an identical situation, advancements in infrastructure and technology have resulted in the emergence of the Big Data issue.

Data can now be generated and collected at extraordinarily high frequencies and quantities because to breakthroughs in sensing, networking, and storage technologies. In only a few hours, large businesses like Facebook, Amazon, and Google create and collect tons of data in the form of event logs or click streams. These data are generally subsequently categorized or picked for individual examination in order to guarantee network security and acquire business information. In personalized online services and recommender systems, for example, web page evaluation must include user session information in order to give better service to each user. Flow design based on network traffic trails would also distinguish various types of network traffic and evaluate evidence accordingly in internet traffic infrastructure. In business transaction assessment, data with certain characteristics is often used for fraud prevention and risk assessment.

A massive quantity of data may be created fast from numerous sources in the age of big data Traditional computer networks are incapable of storing and processing large amounts of data. Cloud computing is commonly used for storing and interpreting large data because of its stable and durable computer capabilities. End-users keep their information in the cloud and depend mostly on cloud server to distribute it with other data consumers via cloud computing. It is important to develop access control methods and confidentiality as per end-user needs in order to just share end-user data with authorized people.

Digital forensics, biology, social sciences, smart grid, and the Internet of Things are just a few of the uses for big data and big data processing According to the estimate given in, a smart grid might create terabytes of data each day from its millions of users at one facility. The knowledge acquired from this massive dataset may be used to improve not only the legacy grid's stability and effectiveness, but also the utility company's consumer contact initiatives. Leakage of confidential user data may be highly harmful to both the company and the individual in question, undermining consumers' trust. On the other hand, ensuring that data may only be accessed by authorized systems or users is critical to maintaining big data confidentiality.

For the objective of analyzing and safeguarding the data on the big data cloud platform, a wide range of approaches have been employed. There have been a few issues with appropriately processing the enormous amount and increasing pace of the data. The scale of the dataset is mirrored in the size of the big data requests, since many requests must be executed in order to sustain and handle the data in the database. This data must be safeguarded to guarantee that it is not tampered with, which can be difficult to recognize and manage efficiently owing to its vast size.

Bilinear Pairing is being used to protect the integrity of the big data in this research paper. This generates pairs of hash keys for the big data that are created at periodic intervals. These frequencies make it easier to keep track on the condition of the data and may be used to spot any anomalies or alterations in the database. The bilinear pairings are used to detect the avalanche effect, which happens when data is even marginally tampered with. This little difference can result in dramatic changes in the Hash keys, which can be discovered and used to show database manipulation. This facilitates for the creation of comprehensive security reports that can be used for forensic data evaluation and tamper detection. This strategy is discussed in further depth in the subsequent subsections of this research paper.

The next step of this research article details the related researches that have been referred in section 2. The section 3 is reserved for the detailing of the presented approach whereas the section 4 lists the various experimental outcomes and the results and the section 5 provides the conclusion and the future directions for research.

2. Literature Survey

K. Yang et. al. [1] suggested an efficient and fine-grained data access management approach for big data that does not leak any private information through the access policy. The suggested solution, unlike previous methods that only partially hide attribute values in access policies, can hide the complete attribute. However, legal data consumers may face significant challenges and difficulties in decrypting data as a result of this. To address this issue, the authors created an attribute localization algorithm that determines whether an attribute is included in the access policy. A unique Attribute Bloom Filter has been designed to locate the precise row to improve efficiency. The authors also showed that their method is selectively secure against specific plaintext assaults. In addition, they developed the ABF using MurmurHash and the access control scheme to demonstrate that their approach can preserve the privacy of any LSSS access policy without imposing a significant burden.

C. Chen et. al. briefly analyzed the cloud-based emergency system, and it was demonstrated that the data upload process during the signing and verification phase is vulnerable. Even though system members' identities are rigorously checked, the attacker can nevertheless monitor a traveler's electronic medical data transmitted via public channels. In addition, if the traveler's mobile device is lost or stolen, the author's approach is vulnerable to an offline password-guessing attack [2]. To address these flaws, they presented an enhanced system that uses the concepts of authenticated key exchange and message authentication to protect traveler privacy. The authors demonstrated that their improved technique achieves the goals of mutual authentication and key agreement using the random oracle model and BAN logic.

L. Watkins et al. suggested a semi-supervised machine learning strategy to dealing with the ever-increasing challenge of Big Data in cybersecurity. They focused on DNS traffic in their analysis, using their method to sift through DNS requests and find the smaller amount of questionable network traffic on the network. The authors accomplished this by using typical clustering methods on a dataset enriched with known harmful domains to filter out the majority of non-malicious network traffic, allowing us to concentrate on a manageable collection of data that most likely contained suspicious or malicious domains [3]. The key to the introduced technique is that it uses DNS

name-based, TTL value-based, and DNS query answer-based behavior of known problematic domains to motivate clustering algorithms. Then, only the inspired clusters are kept, and these become the reduced dataset of dubious findings.

DALM (Dependency-Aware Locality for MapReduce) is presented by X. Ma et. al. for processing real-world input data that is highly skewed and dependent. Data-dependency is accommodated in a data-locality framework by DALM, which naturally synthesizes the fundamental components of data rearrangement, replication, and placement. The authors developed DALM on Hadoop 1.2.1 with Giraph 1.0.0, in addition to algorithmic design. They are particularly interested in the deployment problems, particularly in public virtualized cloud systems. They compare DALM to state-of-the-art solutions, such as the basic Hadoop system, the representative popularity-based Scarlett and the representative graph-partitioning-based Surfer, using both simulations and real-world trials [4]. The results show that the DALM's replication strategy can significantly improve data locality for various inputs; in comparison to the basic Hadoop system, Scarlett, and Surfer, the presented prototype implementation of DALM significantly reduces Hadoop job completion time for popular iterative graph processing apps.

To enhance the result integrity of MapReduce computations, Y. Wang et. al. offered MtMR, a Merkle tree-depend verification system. MtMR uses a hybrid cloud architecture that combines the advantages of both private and public clouds. The public cloud has more computational and storage capacity under this architecture, but it is less trusted. Therefore, while the public cloud can handle the majority of the processing, it cannot guarantee the accuracy of the results. The private cloud is trusted since it is managed by the computing task owner. However, because the private cloud has limited capacity, it can only be used for security-critical tasks. The master and a small number of workers, known as verifiers, are placed on the private cloud in MtMR, whilst the rest of the workers are put on the public cloud [5]. The majority of the work is completed on the public cloud. The integrity of the results is controlled by the master and verifiers on the private cloud. Under the MtMR framework, a semi-honest worker cannot undertake safe cheating and produce good result integrity while incurring a moderate performance overhead, according to quantitative investigation.

To speed up the entire computational process of ELM for huge data, a new SELM algorithm based on the Spark parallel framework was presented by M. Duan et. al.. To speed up the process of decomposing the M-PGIM, the authors proposed a SELM algorithm that consists of three sub algorithms: the H-PMC algorithm, -PMD algorithm, and V-PMD algorithm, which make full use of a series of Spark's good characteristics, such as fault tolerance, persist/cache strategies, partitioning controlled by users, and so on. After that, they used the SELM method to classify large amounts of data. In this research, the authors detailed the process of building the SELM algorithm for big data classification and conducted a performance comparison of their SELM with the competing techniques [6]. Finally, they ran a series of experiments to see how well their SELM performed in terms of medical big data classification and handwritten digit recognition under various settings. The experimental outcome reveals that when compared to PELM, ELM*, and ELM*-Improved, their SELM achieves the largest speedup while maintaining the same accuracy as classic ELM under the same settings.

A. Segatori et. al introduced the DFAC-FFP, a Big Data DFAC based on frequent mining patterns. The MapReduce programming model was created to deal with memory and complexity problems related to AC-FFP by enabling the calculation flux to run in parallel across machine clusters. The DFAC-FFP is divided into three stages: 1) a unique strategy for defining a strong fuzzy attribute distribution based on fuzzy entropy; 2) a distributed FP-Growth algorithm, using the specific strategies used for the production of FCARs in a fuzzy manner; and 3), using the superfluous rules for pruning out superfluous rules, based on fuzzy support, confidence, and distributed training. To achieve their strategy, the authors used the popular framework for Hadoop [7]. Experiment findings in six large datasets show that DFAC-FFP can handle millions of information correctly to learn the AC.

H. Liu et. al. proposed a k-selection of geo-distributed sensitive big data using a distributed privacy-aware rapid selection algorithm. On each server, the algorithm has the worst compute complexity of O (N), which is theoretically the fastest for non-structured data. The number of iterations required to measure communication resource usage is O (log (d/€), where d is the data range length and € is the practical stopping condition. Furthermore, using a properly constructed noise mechanism, the algorithm maintains the privacy of local array statistics in each server to a controllable -differential privacy. Quickselect and optimization-based methods are also distributedly implemented to solve the same problem [8]. A detailed comparison of performance in terms of complexity and privacy guarantee is made with DPAM, demonstrating the latter's great efficiency and privacy-

preserving properties. The authors use simulations and tests to verify the complexity of DPAM in terms of computing and communication efficiency.

S. M. Nabavinejad et. al. introduced a Smart Configuration Selection (SCS) technique for leveraging the power of matrix completion by cautious selecting sample configurations. SCS chooses the most effective sample configurations based on either the Pearson correlation coefficient or the Kendall rank correlation coefficient. The authors test the presented technique on a set of VM layouts already available from cloud suppliers such as Amazon EC2 and Microsoft Azure. They show that the SCS sample configuration selection is close to the best choice and that it can reduce estimation error to 11.58 %, compared to the original 19.72 % of random configuration selection [9]. More importantly, they show that compared to random sample selection, employing SCS estimations in a makespan minimization technique, which employs matrix completion in its profiling phase to estimate the performance of programs on various configurations, reduces execution time by up to 36.03 %.

W. Zhong et. al. introduced BDHDLS to concentrate its efforts on gaining a better understanding of the unique data distribution of individual invasive attacks belonging to specific families. This method is especially useful for detecting subtle data patterns in invasive attacks with a limited number of samples. Both behavioral and content aspects are used by BDHDLS. BDHDLS can analyze intrusive attack samples using both network traffic characteristics and contents in the payload by combining behavioral and content aspects [10]. Because earlier techniques never combined both sorts of features, this tactic can help IDS function better. The simple decision fusion strategy is used in this work to combine the output of different deep learning models in the cluster.

M. Tang et. al. offer three distinct techniques to partitioning input datasets for skyline candidate concurrent computation: i) to overcome the data straggler issue for distributed skyline query processing, they use a simple z-order curve from the literature, (ii) split partitions into separate groups based on approximate skyline candidate distribution among partitions, and (iii) merge multiple partitions into one group based on skyline dominance, to ensure every worker receives a fair proportion of input information and skyline points, and to cut redundant intermediate skyline applicants [11]. By using a data index for finding skyline sets, the authors presented an efficient approach for merging skyline candidates, which improves query processing time by avoiding redundant dominance tests. They implement the recommended method on the Hadoop MapReduce framework. They compare the newly developed approach to previous state-of-the-art algorithms in a large-scale evaluation using well-known benchmarks. The outcome shows that the proposed algorithms outperform state-of-the-art techniques by an order of magnitude.

By revisiting the parallel consistency model, Z. Li et. al. can demonstrate the viability of research in a different direction, namely parallel propagation [12]. Two (parallel) propagation schemes are proposed: static and dynamic, using thread workstation pool and atomic operations to improve propagation efficiency. Additionally, they have improved the STR algorithms to adapt the two parallel propagation schemes, and offer two parallel STR algorithms: PSTRs and PSTRds. The extensive experiments demonstrate that both PSTRs and PSTRds outperform serial propagation in most of the major non-binary table constraints, which show the potential of simultaneous spreading.

S. Wang et al. discussed a variety of issues concerning big data privacy in the context of biomedical research. Because healthcare data frequently contains large-scale clinical and genetic data, which are vast in size and dimension, data privacy is referred to as the "big" component of data privacy. There are some specific obstacles that off-the-shelf tools struggle to deal with. Scalability issues with completely homomorphic encryption and safe multiparty computing algorithms to handle whole-genome sequencing (WGS) data, for example. There are additional difficulties in securing the results of computations on high-dimensional genomic data, which might quickly deplete the budget if not allocated properly [13]. The authors examined cutting-edge privacy-protection technologies for record linkage, synthetic data generation, and genomic data analysis.

The advancement of information technologies has resulted in a global digital transformation across a huge range of industries [14]. Multiple security challenges have arisen as process management becomes increasingly data-driven, using the Big Data Value Chains concept as a data management approach. This level of openness and collaboration necessitates a greater focus on data management. Abou Zakaria Faroukhi et al., emphasize the significance of combining BDVC and security aspects when dealing with security issues. Then, to close the gap among data value chains and cyber-risks, they define appropriate data security dimensions. Furthermore, they present a multi-layered security structure to address security concerns along BDVC. This framework, which is a generic view adaptable for

different domains, enables organizations to protect sensitive data assets and communications across data management processes. This projection ensures a long-term cyber-ecosystem.

Y. Guo et. al. perform a thorough analysis of the functions of big data center applications to effectively improve computer network security protection in the age of big data, to create a comprehensive computational network security protection system. A comprehensive analysis will be made of the hazards associated with modern computer network security, and adequate technologies such as modern network security technologies and systems will be implemented [15] to implement the design of computer network security protection systems in the context of a large-size data era. After the design is implemented, the system is tested accordingly. According to the test results, the computer network security defense system designed can actively detect and prevent security threats in the network, ensuring that the network can operate normally and safely.

3. PROPOSED METHODOLOGY

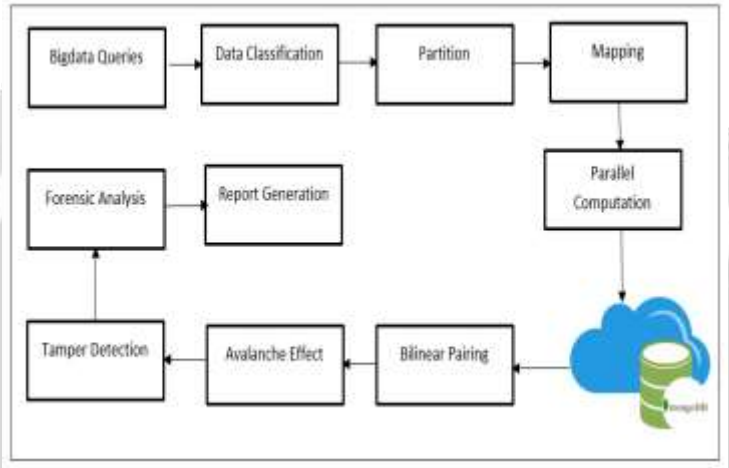

Figure 1: System Overview

The presented technique for the purpose of enabling an effective system for the generation of security report and big data query partitioning and management on the Mongo DB database server has been displayed in the figure 1 above. The process is detailed in a step by step manner below.

Step 1: Query Input and Classification – The presented approach effectively streamlines the query execution procedure for the big data on the Mongo DB database. The big data queries are numerous and can be overwhelming due to the high velocity of the data. Therefore, these queries are classified to achieve useful and effective parallelization for the execution. The input queries are segregated into linear clusters. This allows for division of the queries that can be used for the purpose of parallel execution.

There are a large number of queries are numbered in hundreds therefore, the parallelization considerably improves the performance. The serial execution of these queries would not be possible in real time due to the large size of the data as well as the number of queries. The classification of the queries is performed in this step and then provided to the next step for further execution.

Step 2: Query Partitioning and Mapping – The mapping and the partitioning of the query needs to be performed to ensure a steady flow of the execution. The previously classified queries are therefore, taken as an input in this step of the procedure for the partitioning. The partitioning is an essential aspect of the methodology where the collisions are minimized to reduce the errors. The grouping of the queries can be further performed following the partitioning procedure to achieve greater efficiency.

These groups can be used for the purpose of mapping the respective queries onto threads for the purpose of execution with improved efficiency.

These threads are conceived to achieve the efficient parallel computation which has the ability to speed up the procedure considerably. This is achieved through mapping which can significantly reduce the computational and time complexities of the system considerably. This is useful in achieving an effective improvement in the processing of the big data. The query classification is performed as given by the algorithm 1 given below.

ALGORITHM 1: Query Partitioning

```
//Input : Data Size = SIZE, N = Divisions Required
//Output: C_QL Classified Query List
queryClassificaiton(SIZE, N)
1: Start
2: C_QL =∅
3: D= SIZE / N
4: START=0, END=0
5:   for i=1 to N
6:     T_LIST =∅ [Temporary List]
7:         if ( i==1 OR i<n), then
8:           T_LIST [0]=START
9:           END=START+D-1
10:          T_LIST [1]=END
11:          C_QL = C_QL + T_LIST
12:          START=END+1
13:      end if
14:     else  if ( i==n), then
15:          T_LIST [0]=START
16:          T_LIST [1]=SIZE-1
17:          C_QL = C_QL + T_LIST
18:     end if
19:  end for
20:   return C_QL
21: Stop
```

The partitioning of the queries are performed accurately and is one of the most important implementation in this system. The input queries that are being as an input to this system are effectively partitioned using the partitioning algorithm given above. The total number of queries provided in the input data are subjected to the division by the partition size that is decided earlier.

This leads to a clear number of divisions to be implemented on the input queries. For this purpose, an iteration is initiated from 1 to the number of divisions just achieved. The starting and ending variables are initialized and the limits of the first partition are assigned as zero and the length of the partition reduced by 1 respectively. These values are then appended to a double dimension list and the iteration is continued. The next step the start variable is provided the end variable value and the end values are measured using the partition size, this is done for all the partitions consecutively and the classification of the queries is achieved which is added to the list.

Step 3: Bilinear Pairing – The Mongo DB database is being used for the purpose of storing and retrieving the data. This database needs to be secured against attacks and intrusions and prevent any modifications on the database. Therefore, an effective and useful mechanism for the detection of change in integrity is needed which is achieved through the process of Bilinear Pairing. This is performed through the use of hash key generation of the data stored on the database. This procedure of hash key evaluation is performed at a specific time duration and the hash keys are saved in the database in the form of pairs referred to as the Bilinear Pairs.

The identification of tampering being done on the data being stored on the MongoDB Database is performed through the evaluation of the bilinear pairs. The reliability and security of the database is achieved through the crucial and essential evaluation of the bilinear pairs. The implementation of the Bilinear Pairs evaluation does not put a lot of pressure on the computational complexity and maintains it at a reasonable level. The next step details the evaluation of the integrity of the data using the bilinear pairs generated in this step of the procedure.

Step 4: Tamper Detection and Forensic Analysis – The detection of tampering is one of the most integral concepts in this research. The evaluation of the integrity and its analysis is necessary to understand the evaluation. The pairs from previous step are extracted and then are correlated one pair with another. This is performed to determine if there is any form of modifications that can be evidenced through the presence of the avalanche effect.

Therefore, any type of changes or modifications in the data can lead to a large differences in the hash keys. This presence of the avalanche effect can help us determining that the data is compromised. The Bilinear pairs separated by a time period allow for a better estimation of the tampering performed on the Big Data.

The tampering when identified is then used to alert the user about the tamper and system automatically starts performing a forensic analysis to identify the perpetrator and the specific timing when the tampering act is being performed. The time interval for the tampering must coincide with the time interval of the bilinear pairs during their formation. This and all the other information regarding the tampering is then provided to the user in a security report format that can be effectively used for the purpose of inspection.

4. RESULT AND DISCUSSIONS

The presented technique or the purpose of maintenance and generation of the security reports for the big data management and security has been implemented on the Mongo DB database. The system has been coded in a java programming language using the NetBeans IDE as the standard IDE. The deployment machine contains an Intel Core i5 processor with 6GB RAM assisted by a 500 GB hard drive. The presented approach has been evaluated for its performance through the use of extensive evaluations that are elaborated below.

Performance Evaluation through Root Mean Square Error (RMSE)
The amount of error achieved by the presented technique has the ability to determine the accurate implementations of the methodologies being utilized in this system. For this purpose RMSE is being used to find the error coefficient of the system in achieving the output. In other words, RMSE is the difference between the predicted and actual values. In this way, an effective strategy may be devised, one that can evaluate its RMSE is measured by the following equation 1.

$$RMSE = \sqrt{(xp - xo)}^{2} \quad \text{----------} (1)$$

Where
Xp – Expected number of tuples that are tampered
Xo – Actual number of tuples that are tampered

The suggested system has been subjected to an increasing number of big data queries in order to undertake the experimental assessment. If the queries are executed efficiently, it is feasible to monitor how they are being processed. Every time the experiment is repeated, the number of searches increased considerably. If the system is able to detect the tampering, the predicted detections of manipulation are observed. RMSE measures the difference between predicted and actual tampering detections.
Because the bulk of the tampering was discovered, the RMSE attained by the technique is remarkable. The security report generated as an output enables for the effective realization of security, which can provide robust security to Big Data. The table 1 below is a table of experimental results, which has been used to plot the graph shown in figure 2.

Experiment No	Total amount of Data in the Database	Total amount of Tampered Data	Total amount of detected Tamperings	MSE
1	1000	140	138	4
2	2000	130	127	9
3	3000	110	108	4
4	4000	110	107	9
5	5000	108	106	4

Table 1: RMSE Measurement Table

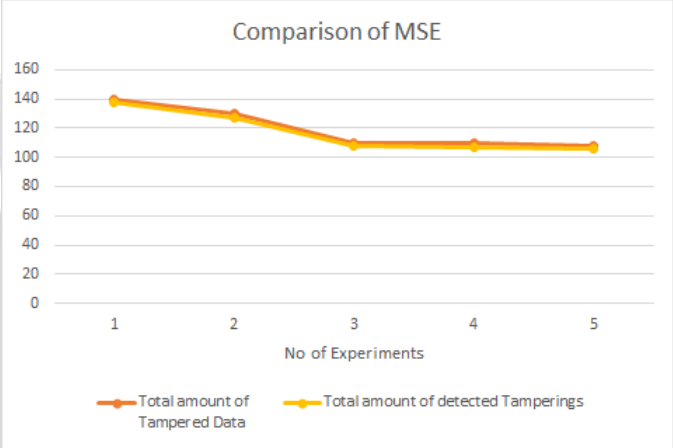

Figure 2: RMSE evaluation Results

With these outcomes and the experimental results, the proposed approach's efficacy has been clearly demonstrated. In terms of error, RMSE has been used to evaluate the approach, and the result is 2.44. This is a very acceptable grade, indicating that the model has been well implemented.

Performance Evaluation for Query Optimization

A test is performed on several datasets obtained from Kaggle and UCI dataset repositories. The experimentation is carried out to determine the time required for the provided number of rows to be inserted into the Mongo DB database in two different scenarios, one without optimization and another with optimization of the query. The values obtained through evaluation of the experiment are shown in Table 1.

DataSet name	No of Rows	No of Attributes	Without Optimization in MS	With Optimzation in MS
MOOC	989	15	2287	690
NHAI	2869	18	2757	1421
UID	911	23	1454	787
Shopping	999	8	1506	987
EEG	507	148	1369	503

Table 2: Optimization Recorded values

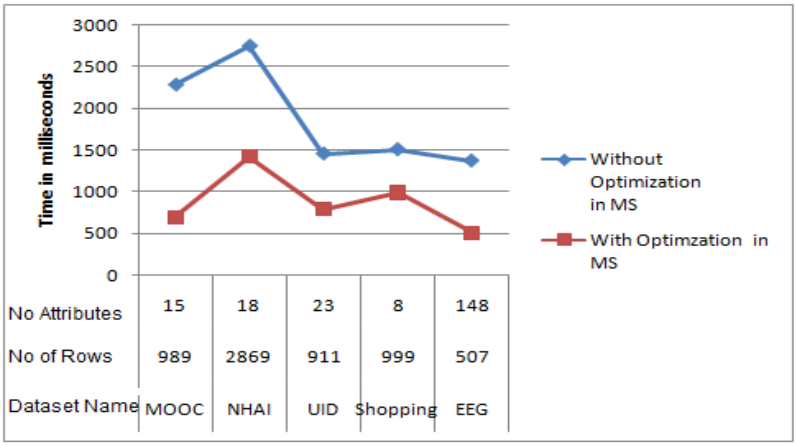

Figure 3: Query Optimization Results for Different Dataset

When executing a certain amount of queries, as shown by figure 3, the suggested model's technique for query optimization has resulted in a far better performance. This experiment clearly demonstrates that the suggested approach reaches a 54 percent Optimization level, which is a great result for an initial deployment of such an innovative methodology.

5. CONCLUSIONS AND FUTURE SCOPE

The presented methodology describes a mechanism for generating effective security reports and managing massive data queries using parallel computing on a mongo DB database. There are usually large-scale data queries on big data that are utilized to efficiently handle and manage the huge data in this technique. As a result of this, the queries are processed and categorized into several sections, such as insert update. These classified queries are effectively partitioned and grouped for mapping purpose. As a result of this mapping, the method may be converted into threads for successive parallel computing. Hashing and bilinear pairing are used to ensure the integrity of the big data. To detect an avalanche effect, pairs of hash keys are created and examined at regular intervals. An analysis report is created and sent to the user after the avalanche effect has been discovered and any tampering on the database has been detected. In-depth experimentation has been performed to realize the system's average error or RMSE as 1.1, which is a reasonable level of error along with acceptable levels of query optimization.

The Future research direction can focus on transforming this methodology into an API for easier integration into existing frameworks and systems to improve the efficiency of Big Data processing.

6. REFERENCES

[1] K. Yang, Q. Han, H. Li, K. Zheng, Z. Su, and X. Shen, "An Efficient and Fine-Grained Big Data Access Control Scheme With Privacy-Preserving Policy," in IEEE Internet of Things Journal, vol. 4, no. 2, pp. 563-571, April 2017, DOI: 10.1109/JIOT.2016.2571718.

[2] C. Chen, C. Li, S. Liu, T. Wu and J. Pan, "A Provable Secure Private Data Delegation Scheme for Mountaineering Events in Emergency System," in IEEE Access, vol. 5, pp. 3410-3422, 2017, DOI: 10.1109/ACCESS.2017.2675163.

[3] L. Watkins et al., "Using semi-supervised machine learning to address the Big Data problem in DNS networks," 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), 2017, pp. 1-6, DOI: 10.1109/CCWC.2017.7868376.

[4] X. Ma, X. Fan, J. Liu, and D. Li, "Dependency-Aware Data Locality for MapReduce," in IEEE Transactions on Cloud Computing, vol. 6, no. 3, pp. 667-679, 1 July-Sept. 2018, DOI: 10.1109/TCC.2015.2511765.

[5] Y. Wang, Y. Shen, H. Wang, J. Cao, and X. Jiang, "MtMR: Ensuring MapReduce Computation Integrity with Merkle Tree-Based Verifications," in IEEE Transactions on Big Data, vol. 4, no. 3, pp. 418-431, 1 Sept. 2018, DOI: 10.1109/TBDATA.2016.2599928.

[6] M. Duan, K. Li, X. Liao, and K. Li, "A Parallel Multi classification Algorithm for Big Data Using an Extreme Learning Machine," in IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 6, pp. 2337-2351, June 2018, DOI: 10.1109/TNNLS.2017.2654357.

[7] A. Segatori, A. Bechini, P. Ducange, and F. Marcelloni, "A Distributed Fuzzy Associative Classifier for Big Data," in IEEE Transactions on Cybernetics, vol. 48, no. 9, pp. 2656-2669, Sept. 2018, DOI: 10.1109/TCYB.2017.2748225.

[8] H. Liu and J. Chen, "Distributed Privacy-Aware Fast Selection Algorithm for Large-Scale Data," in IEEE Transactions on Parallel and Distributed Systems, vol. 29, no. 2, pp. 365-376, 1 Feb. 2018, DOI: 10.1109/TPDS.2017.2761344.

[9] S. M. Nabavinejad and M. Goudarzi, "Faster MapReduce Computation on Clouds Through Better Performance Estimation," in IEEE Transactions on Cloud Computing, vol. 7, no. 3, pp. 770-783, 1 July-Sept. 2019, DOI: 10.1109/TCC.2017.2677906.

[10] W. Zhong, N. Yu, and C. Ai, "Applying big data-based deep learning system to intrusion detection," in Big Data Mining and Analytics, vol. 3, no. 3, pp. 181-195, Sept. 2020, DOI: 10.26599/BDMA.2020.9020003.

[11] M. Tang, Y. Yu, W. G. Aref, Q. M. Malluhi and M. Ouzzani, "Efficient Parallel Skyline Query Processing for High-Dimensional Data," 2019 IEEE 35th International Conference on Data Engineering (ICDE), 2019, pp. 2113-2114, DOI: 10.1109/ICDE.2019.00251.

[12] Z. Li, Z. Yu, P. Wu, J. Chen and Z. Li, "A Novel Multi-Thread Parallel Constraint Propagation Scheme," in IEEE Access, vol. 7, pp. 167823-167835, 2019, DOI: 10.1109/ACCESS.2019.2951027.

[13] S. Wang et al., "Big Data Privacy in Biomedical Research," in IEEE Transactions on Big Data, vol. 6, no. 2, pp. 296-308, 1 June 2020, DOI: 10.1109/TBDATA.2016.2608848.

[14] Abou Zakaria Faroukhi et al., "A Multi-Layer Big Data Value Chain Approach for Security Issues", in 2nd International Workshop on Emerging Networks and Communications, IWENC, 2020, August 9-12, 2020, Leuven, Belgium, DOI: 10.1016/j.procs.2020.07.109.

[15] Y. Guo, B. Zhang and W. Miao, "Research on Network Information Security Protection Technology Based on Big Data," 2020 International Conference on Computer Information and Big Data Applications (CIBDA), 2020, pp. 19-22, DOI: 10.1109/CIBDA50819.2020.00013.