CS-595-10 TERMINATION PROJECT REPORT

QUERY TIME OPTIMIZATION USING HUNGARIAN AND GENETIC ALGORITHM

STEFFY S DIAS

UNDER SUPERVISION OF
PROF. LESLIE LANDER

BINGHAMTON
UNIVERSITY
STATE UNIVERSITY OF NEW YORK

Abstract

The increase in digital gadgets and the use of the internet has exceptionally increased the data on the web, this adversely affects the servers that eventually process the data at their end. To maintain the structured and unstructured data at the server end, NoSQL databases are being used, which can handle all types of data. These NoSQL databases are equipped with the map reducing techniques, even though having map reducing techniques; which handle the data quite efficiently on its arrival in considerable scale, the server often tends to crash due to heavy queue and constant load. This leads to some research on techniques which optimizes the process of data handling and maintaining the queue at the big databases like Hadoop, Apache couch DB , MongoDB and many more. Some existing methodologies are used to apply the optimization technique by clustering the input queries and committing them synchronously reducing the time to commit queries. Some of these methodologies are handling the query cluster data pattern for committing the same. So this paper puts forwards a mechanism of handling the query in an ancient manner to the cluster, then using the KNN which forms due to the hyper graph relation between the important entities. The committing pattern of these clusters is decided by the Genetic algorithm which is powered by a Hungarian model of task handling.

Introduction

Getting insights from available data is a challenging task. In today's era, data collected from various sources is dynamic in nature and present in overwhelming amounts. This data,which has no particular structure, is called Big Data. Volume, variety, velocity and veracity are the four major characteristics of Big Data.

When handling large databases event management is required to maintain the computational limits and at the same time preserve the precious data integrity of the database. As most applications of commercial use concern the use of MySQL as the database and query language of choice for big organisations to a small business owner alike. Therefore, the database is updated and maintained with the help of passing queries in the language of the database, such here it would be SQL or Structured Query Language.

Having a lot of data coming into an organization is one thing and to analyse it is a different thing. To analyse this big data, many data analytic services use Hadoop Distributed File System (HDFS) as a data warehouse and tools provided by the Hadoop Ecosystem. Increasing number of organizations want to have real time insights in order to fully understand what is going on in their organization. However, while analysing this data many problems occur such as higher response time to get the results or insufficient storage resources. As stated above, one of the main characteristics of Big Data is volume, hence, optimizing the time required to obtain results of our queries is a foremost priority. A query optimizer provides a solution to this problem. It is a crucial database management system (DBMS) component that determines ancient execution mechanisms by analyzing the queries. It generates one or more query plans for each query, each of which may use a mechanism used to run a query and the most recent query plan is selected and used to execute the query. In this document, a system is put forth to optimize the query time by using Hungarian algorithm along with Genetic Algorithm and k-nearest neighbour algorithm (KNN). The Hungarian algorithm is an optimization algorithm that solves the problem by assigning the lowest-cost to the jobs so as to minimize the total time of executing all the queries. A genetic algorithm is a search based heuristic algorithm that is based on the process of natural selection where the test individuals of the current generation are selected for reproduction in order to produce offspring of the next generation. KNN algorithm can be used for classification as well as regression. It is a lazy learning, non-parameterized and

supervised learning algorithm that uses a dataset in which the data objects (data points) are separated into various classes and the algorithm predicts the class of a new data object. This algorithm predicts the class out of the new data object by a maximum vote of its neighbors.

For handling a large enough database or making a lot of edits or updates to a reasonably sized database would normally require a large amount of memory and that would generate a substantial computational overhead. A complicated enough query can even freeze the system until it completes and that would render the database useless. Therefore, all the queries are handled by the Query Optimiser. It is a mechanism by which the queries read are managed efficiently while reducing the load on the database.

The Query optimiser follows a set of rules, called Query Plans that help it make an efficient decision regarding the query. It basically reduces the load on the resources while it's being executed to not put undue load on the server. The query also prioritises the Queries according to the Query plan and executes them accordingly. The Query Optimiser executes its function independently and it is invisible to the user. It is a highly efficient and useful part of a SQL Database.

Hungarian algorithm is one of the oldest methods for optimisation problems. It was first developed in 1955 and is still widely in use today. It is a Combinatorial Optimisation Algorithm, that is used to solve various Assignment problems. It is a highly efficient and useful algorithm to assign the jobs by matching the elements with the lowest cost. The hungarian algorithm follows certain steps that are essential for its working. It has been famous ever since and is still in widespread use to this day. The method was given the name due to the immense dedication and work performed by the scientists that were Hungarian. Therefore, to honour their dedication, it was named the Hungarian Algorithm. The Hungarian Algorithm solves the assignment problems with the help of minimum matching that is observed in the data elements. The Hungarian Algorithm is a very simple solution to the assignment problem and one of the most useful. The Algorithm is also highly powerful and generates extremely accurate results that can be applicable in real time.

Genetic Algorithm is one of the powerful algorithms to perform searches faster and efficiently. It is a heuristic method for the searching of elements in large datasets. It is one of the most important features for application in Artificial Intelligence. Genetic Algorithms are also used in applications concerning

optimisation problems as they can execute in unconstrained or constrained environments therefore, they're quite flexible for the deployment. It is a highly efficient and powerful algorithm that can provide consistent and stable results as it is less likely to use breakdowns due to its inherent robustness when compared to other conventional methods that are used for solving optimisation problems.

Motivation

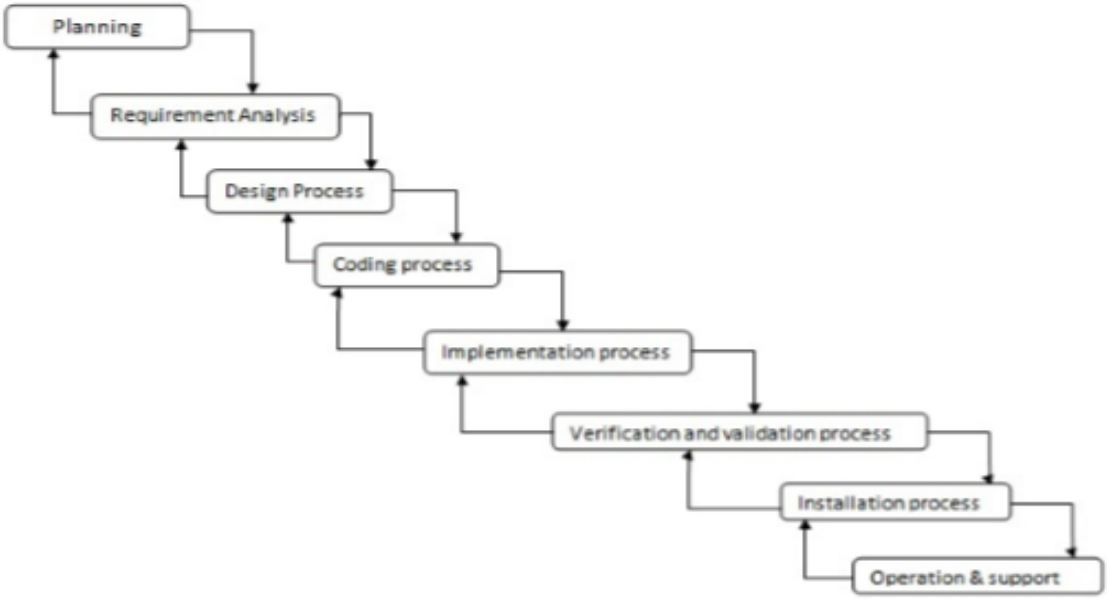
A major reason so as to why a query does not perform well is when there is a large amount of data; the queries have to sift through the whole lot to give results. Queries for big data must therefore be carefully coded in order to make efficient usage of the system resources and save time. Time is one of the greatest assets in software engineering. With a drastic development in the data analysis workflow, the analysis of data has become faster and easier. This in turn helps in faster decision making by saving time. Therefore, the proposed idea of building a system that converts data into graphs and further into hyper graphs, by using Hungarian Algorithm along with Genetic Algorithm, the system achieves faster transaction time.

Problem Definition

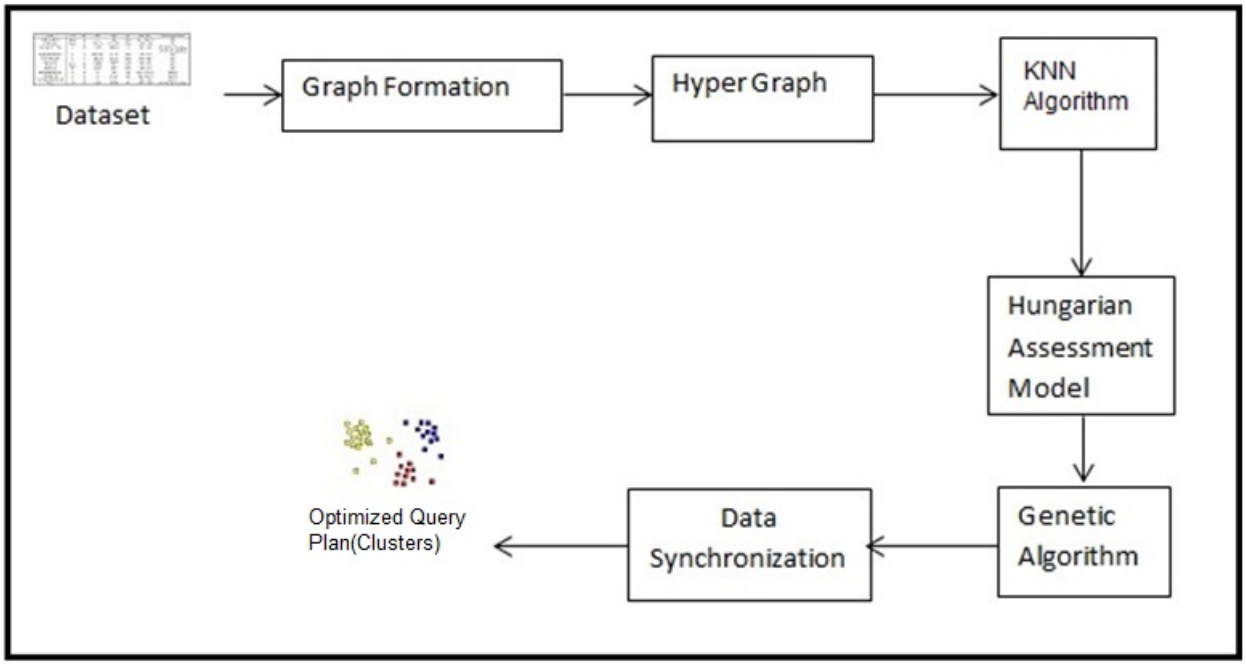
To optimize the query processing time by decomposing the data into hyper graphs based on correlating features and then using Hungarian Algorithm and Genetic Algorithm for its processing.

System Design

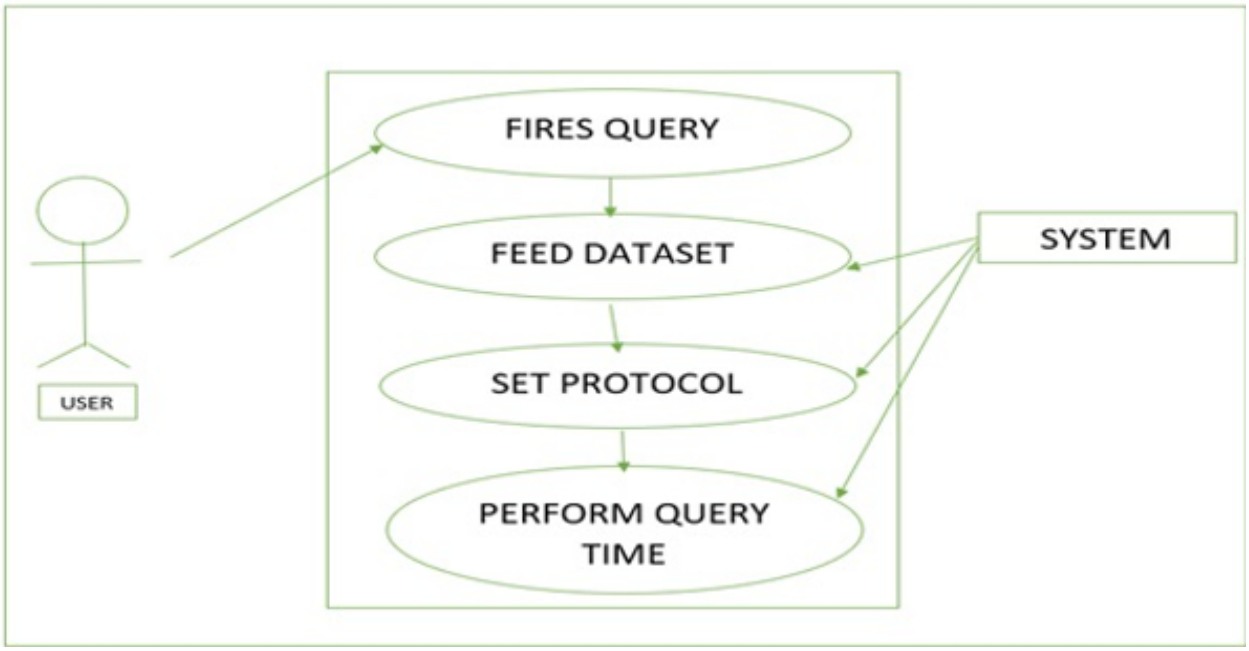
System Implementation Plan



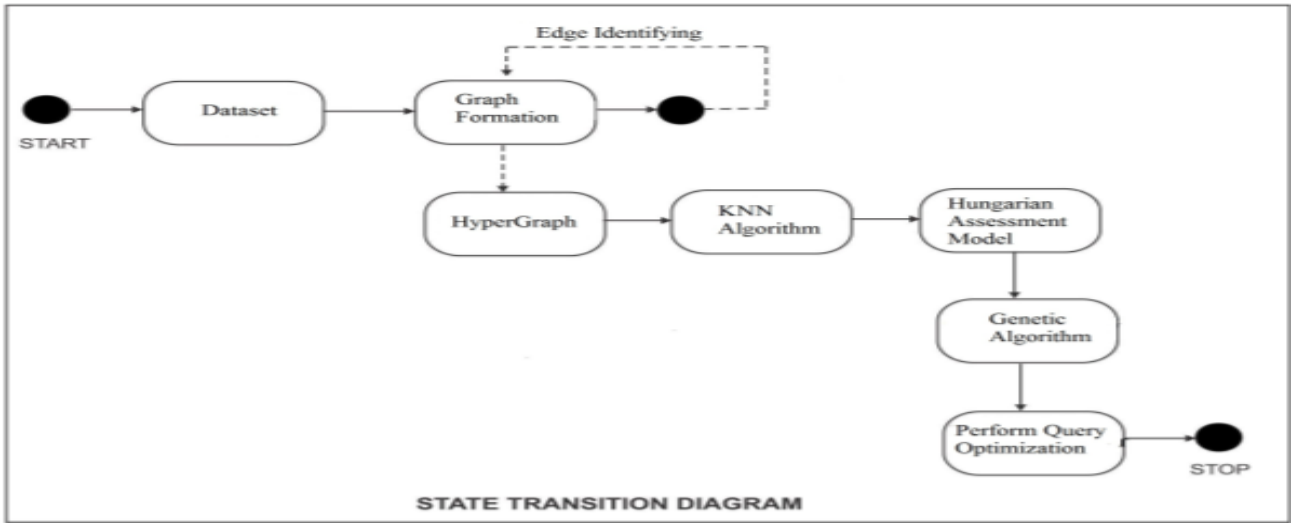
System Architecture



Use Case Diagram



State Transition Diagram



Implementation

A system that transforms the data into graphs to give it a sense of structure and stores them in graph-based scalable stores like neo4j. Storing data in the form of graphs allows us to stream changes dynamically. The structure of these graphs is in the form of node-predicate-node triples. The graph triples will be converted into quadruples by using chromosomeID as an extra eld to represent which node belongs to which population using genetic algorithm. Thus, the structure of the quad will become chromosomeID-node-predicate-node. This chromosomeID represents annexure of the cluster and not a member of the newly formed quad. The quadruples that belong to the same cluster and have a high degree of connectivity are placed into the same partition to ensure locality of intra-cluster quadruples. Closely connected clusters are placed in nearby partitions physically by comparing the inter-cluster distance with the physical distance of partitions. The number of partitions depends upon the number of machines; each machine has its own partition. The quads related to any random cluster are placed in the same partition. This task is performed by Map Reduce by scanning all the quadruples. In the second scan, the quadruples with the closest inter-clusters are placed in the same partition and released from the original dataset. Further, the second closest inter-clusters are placed into the next closest partition. This process is put in iterative mode till no more interrelated clusters are left. Queries are red on these clusters. Hash partitioning is used to map the same subjects to the same blocks which results in faster throughput. To optimize our query time even further, we schedule the queries depending on the time taken to solve each query. Multiple query plans are generated using Hungarian algorithms and the one which results in minimum execution time is selected for execution. Thus, our system smartly detects, distributes and manages data over a scalable and distributed le system.

Step 1:

Dataset collection and Hyper graph Generation- This is the initial step of the proposed model where many datasets are collected for the module deployment. These datasets are collected through the huge dataset repositories like UCI and Kaggle. These Datasets are stored in the workbooks with specific attributes in rows and columns.

Step 2:

KNN Clustering- KNN is a non-parametric supervised learning technique in which we try to classify the data point to a given category with the help of a training set. In simple words, it captures information of all training cases and classies new cases based on a similarity.

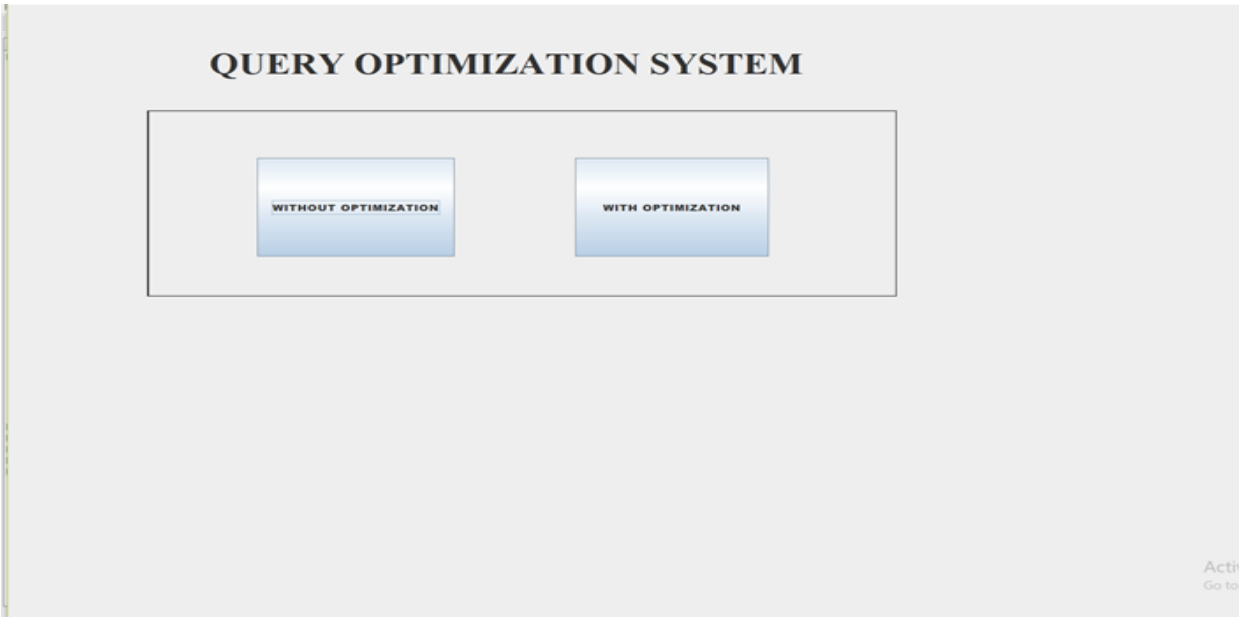
Step 3:

Hungarian Genetic Algorithm -The Hungarian method is a combinatorial optimization algorithm that solves the assignment problem in polynomial time and which anticipated later primal-dual methods.

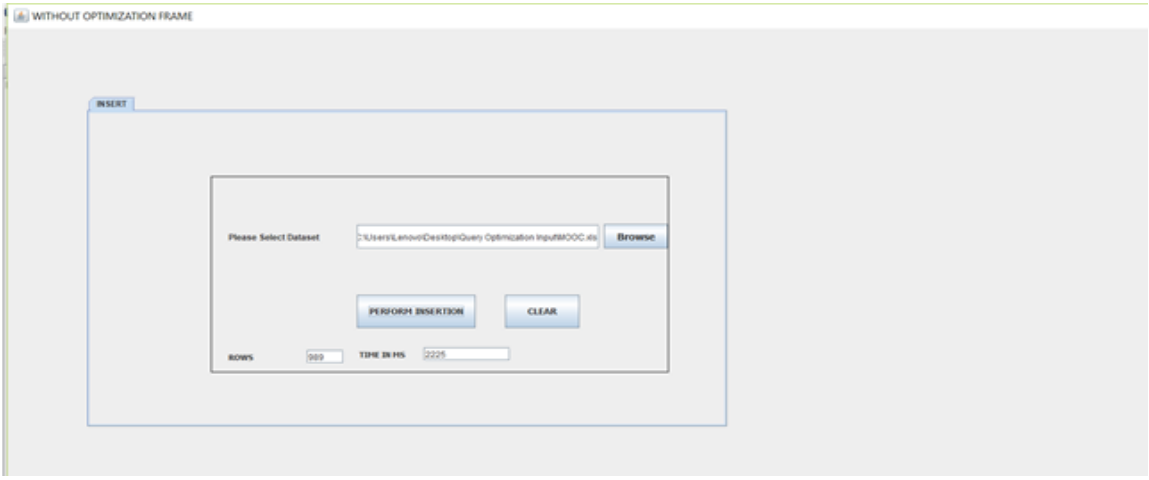
Step 4:

Data Synchronization - Here in this step the clusters are selected based on the obtained pattern numbers from the Hungarian Genetic model. Then these clusters are loaded to the respective threads to query commitment. This leads to optimized time taken to commit the queries of huge numbers into the NoSQL database MongoDB.

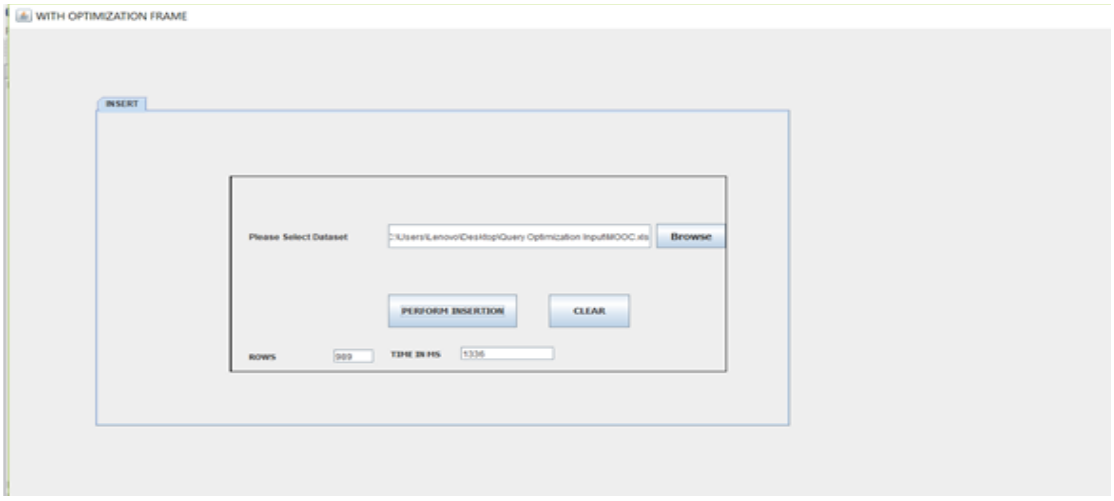
Implemented Project Screenshots



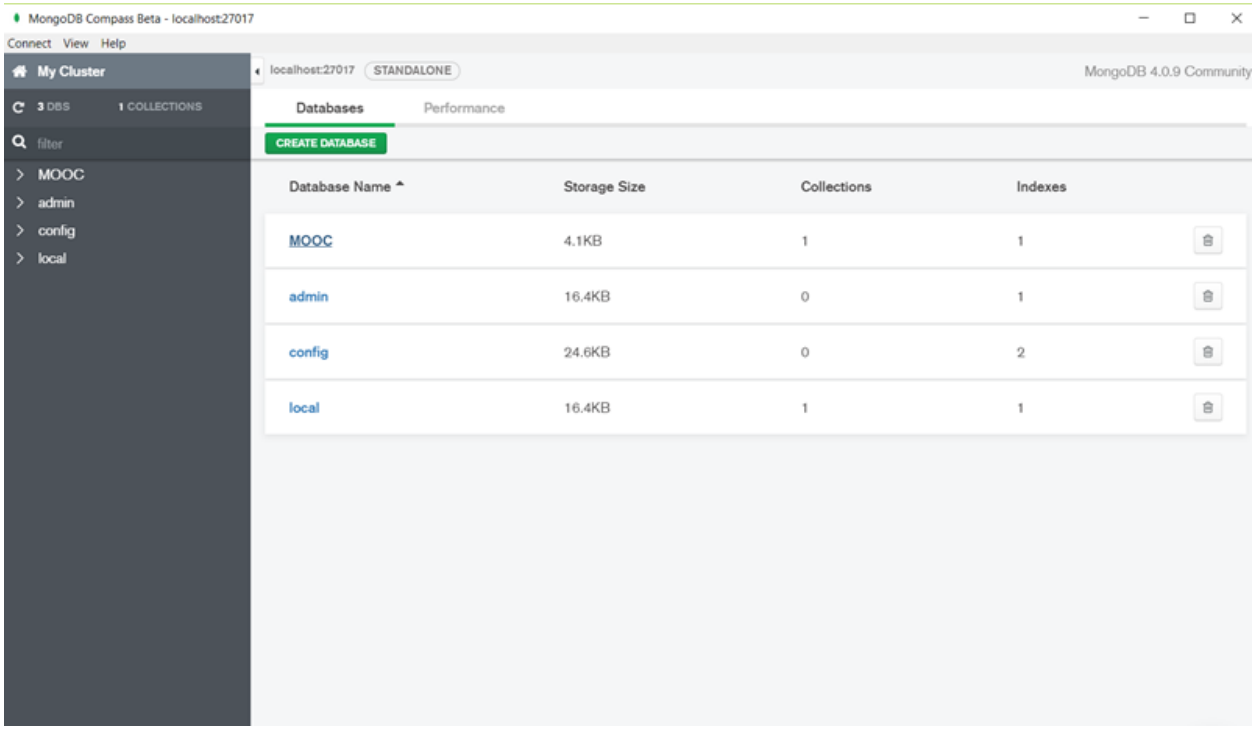
Mainframe



With Optimization Frame



Without Optimization Frame



Database Storage

Result

Result and Graph

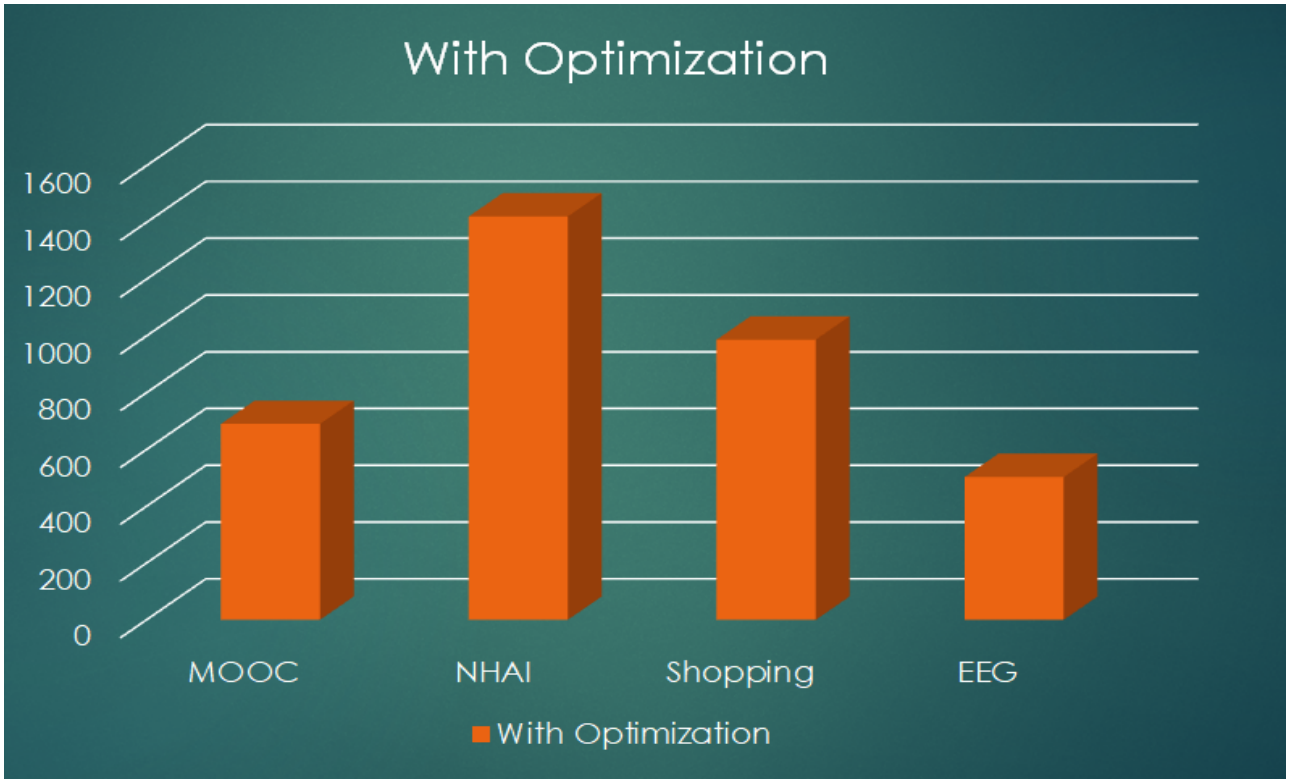
The basic idea of this project comes from the fact that many times, when huge data is committed in a database, it takes more time than estimated which always creates trouble for database transactions. To make this process easier, data needs to be clustered based on the common attributes so that the process of data committing in the database can occur faster. To achieve this, a database like Hadoop has a structure of Map-Reduce. Most of the time, this is not enough to handle the process smoothly. As a result, huge accumulating transactions occur which eventually makes the process slower and adversely affects the performance of the application which in turn creates havoc at the user end.

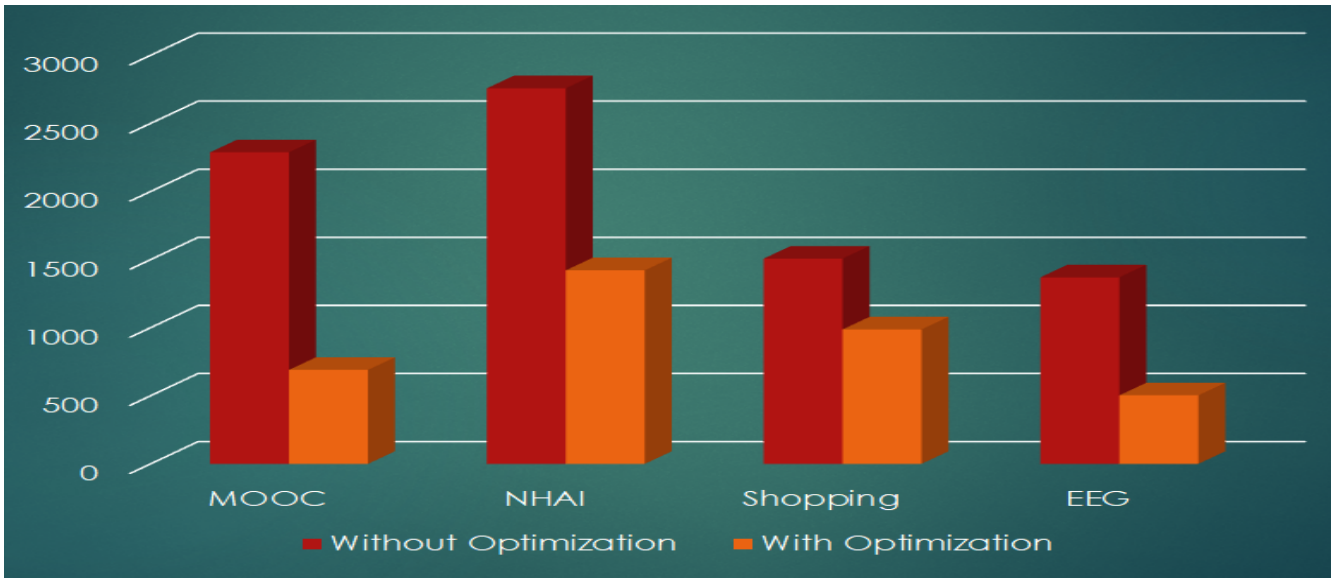
A major reason as to why a query does not perform well is that when we have a large amount of data, the queries have to sift through this whole lot to give us the correct results. Queries for big data must therefore be carefully coded in order to make efficient usage of the system resources and save time. Time is one of the greatest assets in software engineering. With a drastic development in the data analysis workflow, the analysis of data has become faster and easier. This in turn helps us in faster decision making by saving time. We have therefore proposed the idea of building a system that converts data into graphs and further into hyper graphs. By using Hungarian Algorithm along with Genetic Algorithm, the system will achieve faster transaction time.

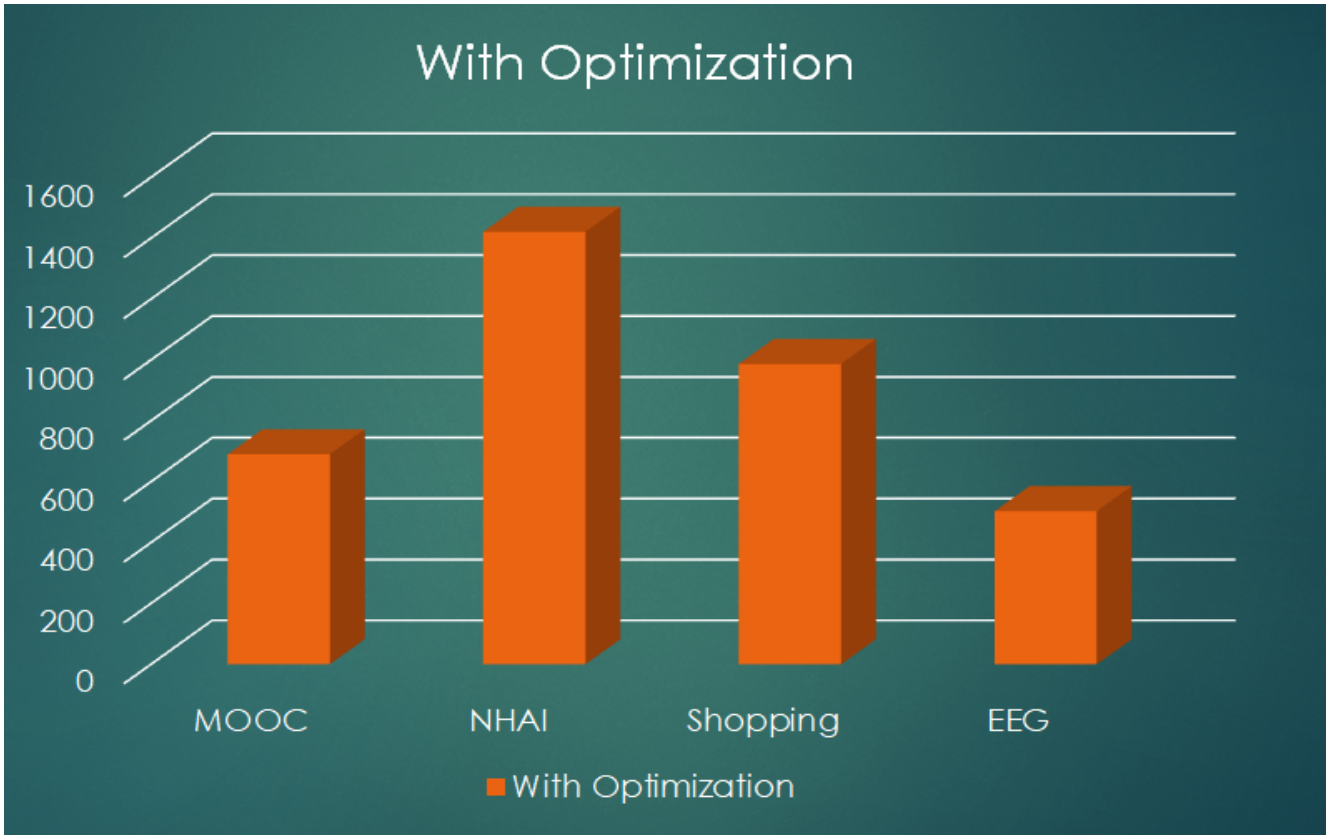
Therefore optimize the query processing time by decomposing the data into hypergraphs based on correlating features and then using Hungarian Algorithm and Genetic Algorithm for its processing. To develop the model of query optimization in NoSQL windows laptop is used, powered by Intel Pentium core i5 Processor along with the 6GB Primary memory. To develop the model the Java programming language is being used with Netbeans as the development IDE. And the proposed system uses the MongoDB NoSQL database for the experimental setup. Our system interacts with the secondary storage device while reading the data from the dataset. The model is not using any external API's or tools so it interacts only with the Hadoop database server while reading some stored protocols.System should process requests to optimize the queries even with minimum processor speed. Then system must wait for process completion.System should correctly execute the

process, display the result accurately. System output should be in user required format. That means all the intermediate steps need to be displayed properly.

Dataset name	No of Rows	No of Attributes	Without Optimization in MS	With Optimization in MS
MOOC	989	15	2287	690
NHAI	2869	18	2757	1421
Shopping	999	8	1506	987
EEG	507	148	1369	503




3000
2500
2000
1500
1000
500
0
MOOC
NHAI
Shopping
EEG
Without Optimization
With Optimization


With Optimization
1600
1400
1200
1000
800
600
400
200
0
MOOC
NHAI
Shopping
EEG
With Optimization

Other Specification

❖ Advantages

- Reduces transaction time
- Works in less space complexity

❖ Limitations

- Can handle moderate amount of Query for the considered machine
- Speed depends on the IDE versions

❖ Applications

- Web applications where heavy database transactions need to happen like banking, Telecom and other sector

Conclusion and Future Scope

Conclusion

Database query committing time optimization is the need of the hour as most of the SQL databases are not equipped with the map reducing techniques. Whereas NoSQL databases are equipped with the map reducing technique, but this map reducing is available purely for the internal management of the threads or for multi tasking. Even this map reduction is not sucient for the vast incoming queries at the database servers. So this paper clusters the incoming queries by using the K Nearest neighbour clustering technique based on the hyper graph correlation scheme. These clusters are evaluated for their committing patterns in the database using the Hungarian genetic algorithm to enhance the process of optimization. The obtained patterns from the Hungarian and Genetic algorithm are used to load the clusters into the respective threads to commit the query successfully and thereby to reduce the time complexity. The experimental results show that the proposed model achieves almost 54% of optimization rate that is really quite good performance. Therefore the proposed system efficiently identifies the correlated features of the transaction data using the hyper graph due to this data being segregated semantically rather than based on the attributes. Once the Data is segregated then their clusters are created using K nearest neighbour algorithm to form more nice clusters. The generation of classiers prepared by Genetic Algorithm and the assignment of queries through Hungarian Algorithm will lead to optimized query plans.

Future Scope

In the future the query optimization can be applied in real time cloud data centers to handle the data of size in GB using the distributive paradigm, also the system can be authenticated and authorized to specific users as well as system can be built as a ready made API for web service platforms.

References

- Mustafa Hajeer , Dipankar Das Gupta œHandling Big Data Using a Data-Aware HDFS and Evolutionary Clustering Technique, IEEE Transactions On Big Data,2017.
- G. Yildirim, I. Hallac, G. Aydin and Y. Tatar, œRunning Genetic Algorithms on Hadoop for Solving High Dimensional Optimization Problems, 9th International Conference on Application of Information and Communication Technologies (AICT), 2015.
- Y. Shen, C. Hsuand, S. Hsieh, œIntegrated Genetic Algorithms and Cloud Technology to Solve Travelling Salesman Problem on Hadoop, IEEE 4th International Conference on Cloud Computing Technology and Science, 2012.
- X. Huang, H. Zhou and W. Wu, œHadoop Job Scheduling Based on Hybrid Genetic Algorithm, International Conference on Cyber Enabled Distributed Computing and Knowledge Discovery, 2015.
- L. Shouqiang and Q. Ming, Research and Design of Hybrid Collaborative Filtering Algorithm Scalability Reform Based on Genetic Algorithm Optimization, 6th International Conference on Digital Home, 2016.
- J. Liu and W. Chen, œFATDOG: Hadoop based Test Data Generation Tool for RESTful Web Service, IEEE 41st Annual Computer Software and Applications Conference, 2017.
- M. Mizukoshi and M. Munetomo, œDistributed Denial of Services Attack Protection System with Genetic Algorithms on Hadoop Cluster Computing Framework, IEEE Congress on Evolutionary Computation (CEC), 2015.
- L. Geronimo, F. Ferrucci, A. Murolo and F. Sarro, A Parallel Genetic Algorithm Based on Hadoop MapReduce for the Automatic Generation of JUnitTest Suites, IEEE Fifth International Conference on Software Testing, Verification and Validation, 2012.
- B. Chang, H. Tsai, Z. Lin and C. Chen, Access Security on Cloud Computing Implemented in Hadoop System, Fifth International Conference on Genetic and Evolutionary Computing, 2011.
- P. Sachar and V. Khullar, Social Media Generated Big Data ClusteringUsing Genetic Algorithm, International Conference on Computer Communication and Informatics, 2017.