Parallel and Distributed Computing

Review - 3

PARALLEL DISTRIBUTED ALGORITHM FOR RAILWAY SIGNAL FAULT DETECTION

Project done under the guidance of Prof. Narayanan Prasanth

Members :

Arshia Shetty - 17BCE0076

Nikila GS - 17BCE0272

Priyanshu Panda - 17BCI0085

Sagar Suman Dwibedi - 17BCI0098

Anish Nagesh - 17BCI0192

DECLARATION

I hereby declare that the report entitled "PARALLEL DISTRIBUTIVE ALGORITHM FOR RAILWAY SIGNAL FAULT DETECTION" submitted by Priyanshu Panda - 17BCI0085, Sagar Suman Dwibedi - 17BCI0098, Anish Nagesh - 17BCI0192, Arshia Shetty - 17BCE0076 and Nikila GS - 17BCE0272, for the course CSE4001 Parallel and Distributed Computing (EPJ) to VIT is a record of bonafide work carried out by us under the supervision of Dr.N.Narayanan Prasanth.

I further declare that the work reported in this report has not been submitted and will not be submitted, either in part or in full, for any other courses in this institute or any other institute or university.

Place : Vellore

Date : 2/06/20

Signature of the Candidate

ACKNOWLEDGEMENT

We would like to specially thank Dr.N.Narayanan Prasanth, School of Computer Science and Engineering, whose immense guidance helped us to complete this project. We wish to express our sincere thanks to Honourable Chancellor, Dr.G.Viswanathan; respected Vice-chancellor, Dr. Anand Samuel of this prestigious college for giving us an excellent world class academic environment and facilities for pursuing our degree. We would also like to thank our friends for their timely help and suggestions rendered for the successful completion of this project. This acknowledgement would be incomplete without expressing our whole hearted thanks to our parents for their continuous support and guidance.

ABSTRACT :

To explore the data processing of high-speed railway fault signal diagnosis based on MapReduce algorithm, the partitioning of the data set has been improved and implemented using K- means clustering algorithm. After which SON algorithm and Markov models have been applied to get better and improved results. In MapReduce parallelization process, the data partition matrix Tk was stored in line segmentation, the computing load was distributed in every node of cluster, and the time consumption of mobile data matrix and the consumption of partitioned matrix were calculated.

List of Figures

List of Tables

List of Abbreviations

DTMC	Discrete Time Markov Chain
TF-ID	Term Frequency − Inverse Document Frequency
HTTP	Hypertext Transfer Protocol

INTRODUCTION :

Objective:

The main objective of this project is to develop an efficient system such that we will be able to minimize the accidents that occur on a regular basis on the railway tracks. The main aim of this project is to reduce the unfortunate events that cuase damage to people and the government.

Motivation:

With the increase of speed of India's high-speed railway and the continuous improvement of the railway information system, the conditions of collecting more railway running information are now available. At present, the running high-speed railway train, through the deployment of a large number of sensors, collects a variety of data. However, the traditional vibration data feature extraction and analysis technology is running on a single machine. This kind of technology, in the mass vibration data acquired by sensors, exposed the shortcomings of long processing time, various artificial intervention, and poor capability of processing big data file and so on. The emergence of cloud computing technology provides a way of thinking to solve the above problems. Map Reduce is an effective parallel computing framework of processing big data, which is one of the main models of cloud computing, and can automatically assign tasks and realize task balance. The working principle, operating mechanism and fault tolerance mechanism of Map Reduce calculation model are studied. In addition, combined with the characteristics of association rule generation algorithm, the traditional parallel algorithm is improved and the parallel optimization scheme of association rules algorithm based on Map Reduce is proposed.

LITERATURE SURVEY :

AUTHOR	PROBLEM STATEMENT	METHODOLOGY	EVALUATION METRIC	RESULT	LIMITATION
Optimization of a genetic algorithm for road traffic network division using a distributed/parallel genetic algorithm By Tomas Potuzak **Published in:** 2016 9th International Conference on Human System Interactions (HSI) **Date:-** 8th July, 2016	Computer simulation of road traffic is an important tool for analysis and control of existing and projected road traffic networks. Such a simulation can be very time-consuming for large road traffic networks (e.g., big cities or even states), therefore many road traffic simulators should be performed in a distributed computing environment.	A distributed/parallel genetic algorithm **(DGA)** for the optimization of the parameters settings of the genetic algorithm for the road traffic network division. Necessary to divide the simulated road traffic network into sub-networks, whose simulations are then performed as processes on individual nodes of the distributed computer.	**The parameters include:-** 1. Number of DGA generations. 2. Number of DGA parent individuals. 3. The number of mutations per DGA individual 4. Type of DGA selection 5. Type of DGA crossover.	In order to determine, whether the new DGA settings obtained from the OGA is better than the original DGA settings, we performed a set of tests on five road traffic networks - three regular square grids, which were already used by the OGA. As expected the new DGA settings gave better results.	1. OGA computation was regularly interrupted during test traffic. 2. Production of irregularly high mutations could cause negative results. 3. However, the better results are for the price of far higher DGA computation time. With the new settings, the DGA is on average 185 times slower than with the original settings.
Research of an Improved Apriori Algorithm in Data Mining Association Rules By Jiao Yabing Published In: International Journal of Computer and Communication Engineering, Vol. 2, No. 1, January 2013 [17BCI0098]	Apriori algorithm is the classic algorithm of association rules, which enumerate all of the frequent item sets. When this algorithm encountered dense data due to the large number of long patterns emerge, this algorithm's performance declined dramatically. This paper illustrates about enhancing the algorithm for better performance	In the Apriori algorithm, C_{k-1} is compared with support level once it was found. Item sets less than the support level will be pruned and L_{k-1} will come out which will connect with itself and lead to C_k . The optimized algorithm is that, before the candidate item sets C_k come out, further prune L_{k-1}, count the times of all items occurred in L_{k-1}, delete item sets with this number less than k-1 in L_{k-1}. In this way, the number of connecting items sets will decrease, so that the number of candidate items will decline.	This study is focused on how to solve the efficient problems of Apriori algorithm and raise another association rules mining algorithm. This has certain reference value to research and solve the issues of data expiation and information lacking.	This decreases the possibility of combination, decline the number of candidate item sets in C_k , and reduce the times to repeat the process. For large database, this algorithm can obviously save time cost and increase the efficiency of data mining. This is what Apriori algorithm do not have.	Although this process can decline the number of candidate item sets in C_k and reduce time cost of data mining, the price of it is pruning frequent item set, which could cost certain time. For dense database (such as, telecom, population census, etc.), as large amounts of long forms occur, the efficiency of this algorithm is higher than Apriori obviously
A novel fuzzy logic controller based automatic caution order system for Indian railways By K.P Ajeesh R Ashis. Mary Helna Muttath Nidheesh.K. Chandran C.R Sreedevi **Published in:** 2012 IEEE International Conference on Advanced Communication Control and Computing Technologies (ICACCCT) **Date:-** 04 October 2012	This paper focuses a new way of approach to find the solution for the artificial intelligent braking system in train using the fuzzy logic controller. The new way of approach is to reduce the man power and to automate the system.	The main function of the fuzzy logic controller used here is to automatically stop the train in each station without any manual procedure of stopping the train. The fuzzy logic controller in train gets activated about 500m from the station so that the train stops at the station smoothly and automatically. The fuzzy controller takes the decision with reference to the speed and distance of the train.	1. **Crisp Input and Output Map in Matlab**:-The fuzzy predictive control technology is compared with the conventional control technology. The simulation results show that the performance of train safety, comfort, parking accuracy and other performance indicators have been improved significantly by using fuzzy predictive controller.	In the simulation circuit the speed of the motor is one input of the fuzzy logic controller, another input is distance. Distance is calculated from the speed. The fuzzy logic controller output is given to the motor. This is the test circuit of the fuzzy logic controller. So the motor stops when the distance becomes zero.	1. It is tedious to develop fuzzy rules and membership functions and fuzzy outputs can be interpreted in a number of ways making analysis difficult. 2. In addition, it requires lot of data and expertise to develop a fuzzy system.

Comparative Analysis of Map Reduce Scheduling Algorithms By Sonia Sharma, Dr. Parag Jain. Published In: Jour of Adv Research in Dynamical & Control Systems, Vol. 11, 10-Special Issue, 2019 [17BCI0098]	When Hadoop schedules reduce tasks, it neither exploits data locality nor addresses partitioning skew present in some MapReduce applications. Scheduling of mixed real-time and non-real-time applications in MapReduce environment is a challenging problem but accepts only restricted attention. First In First Out (FIFO) is the default job scheduling strategy of Hadoop, but it cannot guarantee that the job will be completed by a specific deadline.	In this work they have compared the various scheduling algorithms and their counterparts using MapReduce framework. They have developed FIFO, Round Robin, ETF and LATE schedulers with MapReduce compatible implementatio n of the same namely MR-FIFO, MR-RR, MR-ETF and MR-LATE.	To calculate the performance gain the one algorithm for which the value is calculated is subtracted the reference algorithm, divided by the gain by the original algorithm turnaround time. Finally, multiply it by 100 to get the percentage change.	The MapReduce framework along with the overview of different scheduling techniques with their applications is discussed in this paper. Different scheduling techniques to enhance the data locality, make span, efficiency, fairness and performance are discussed with special emphasis on its real time applications. This work they have compared the various scheduling algorithms and their counterparts using MapReduce framework and developed FIFO, Round Robin, ETF and LATE schedulers with MapReduce compatible implementation of the same namely MR-FIFO, MR-RR, MR-ETF and MR-LATE. Classic benchmarks of Max of Eigen Vectors was used to evaluate the performance of our scheduling algorithm. For each test every benchmark is run for 100-1000 tasks and the average execution time is used as the result.	MR-FAIR scheduler needs to be improved in using (1) Application of Optimization algorithm to map reduce (2) Application of non-uniform task distributions. (3) Trying to enable the optimization based algorithm to schedule real time jobs.

Parallel computing in railway research By: Qing Wu, Maksym Spiryagin, Colin cole & Tim McSweeney Date: Dec 2018	Parallel computing applications in railway research as well as the enabling techniques used for the purpose.	one publication can use multiple enabling techniques. For example, the Domain Decomposition Method (DDM) technique has to be used with another communication technique such as the MPI to successfully conduct parallel computing	Iterative optimisations, such as Particle Swarm Optimisations (PSO), Genetic Algorithm (GA) and Simulated Annealing (SA), improve a group of solutions (plans, designs, etc.) iteratively in loops. Almost all iterative optimisation methods have algorithm structures that are very suitable for parallel computing, especially during the fitness assessment step	parallel computing applications and enabling techniques that are used in railway research were reviewed. Nine enabling techniques were reviewed and MPI, DDM. and Hadoop & Apache are the top three most widely used enabling techniques. Seven major application	(1) the engineering basis using clusters for parallel computing was arguably invented by Gene Amdahl in 1967 (2) PCs were only made available in the late 1970s (3) the MPI and OpenMP standards were only

DISTRIBUTED COMPUTING IN MOBILE ENVIRONMENT By Apekshit Sharma, Chandrakant Sharma	The classic models and solutions are not adapted to mobile world's hence new models; new problems and obviously new solutions should be designed. This paper advocates for the creation of a common view for mobile worlds based on the similarities between them.	The RL algorithm assumes that there is a unique token initially and utilizes the partial reversal to maintain a token oriented DAG (directed acyclic graph). A node should gain the token along the DAG to access the shared resource. The proposed algorithm in this paper is also adapted from the RL algorithm. The RL algorithm and its adaptations are sensitive to link forming and link breaking.	State: Indicates whether node i is in the ES, CS, or NCS state. Initially, state = NCS. N: The set of all nodes (neighbors) in direct wireless contact with node i. Initially, N contains all neighbors of node i. hVector: An array of triplets representing node is view of height of node j, j € N. Initially, hVector[j] = height	1. We propose a distributed algorithm for solving the h-out of-k mutual exclusion problem for ad hoc mobile networks. 2. The proposed algorithm is sensitive to link forming and link breaking and thus is suitable for ad hoc mobile networks. 3. This paper pointed out similarities among common mobile systems	1. The nodes may not have unique node identifiers. 2. Node failures may occur. 3. Communication links can be unidirectional and not FIFO. 4. Incipient link failures may not be detectable. 5. Message delays don't obey the triangle inequality

Parallel and distributed kmeans to identify the translation initiation site of proteins **By:-** Laerte M. Rodrigues ; Luis E. Zárate ; Cristiane N. Nobre ; Henrique C. Freitas **Published in:** 2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)	Use of Parallel kmeans algorithm to enhance performance of TIS and non-TIS class balancing for databases where the imbalance is approximately 1:40.	The strategy is based on MPI (Message Passing Interface) and OpenMP (Open Multiprocessing) to exploit shared memories, and collective communications to reduce the influence of network. This way, the proposal is to achieve efficiency and scalability.	1. The number of databases. 2. The splitting of databases and clustering. 3. Stability tendency of centroids.	The processing time gain per iteration demonstrates a high speedup, indicating that the solution considerably reduces processing time and guarantees a response in a much lower time.	1. The efficiency indicates that the processing cores are being underused. 2. For small databases the small number of operations reduces the efficiency. 3. The time taken for all of the databases is directly related to the quantity of clusters that should be formed. Which cause longer processing.

A quick Otsu-Kmeans algorithm for the internal pipeline detection **By:-** Tao Song ; Ji Li ; Xue Han ; Lei Shao **Published in:-** 2017 IEEE International Conference on Mechatronics and Automation (ICMA)	The traditional kmeans algorithm is sensitive to the initial clustering center and the slow processing speed of large data set. It is adopted to make the image partition processing and using the otsu algorithm to find the right clustering center.	Aiming at the defects of the pipeline, this paper proposes an improved kmeans clustering algorithm for image segmentation. First of all, according to the characteristics of image processing, the original image were divided into blocks and the sub block variance is greater than 10% of the original variance. Then, image segmentation is performed on this sub block by using Otsu, and the two kind of mean value is obtained as the initial clustering center of kmeans	1. Value of Segmentation threshold. 2. Gray level of the image 3. RGB level of the image.	The experiment shows that the clustering effect in this paper is much better than Otsu method based on threshold and region growing method based on region, and the speed is improved than the traditional k-means algorithm. Pipeline robot has an important value in inspecting small pipes, and effective processing the collected images is key to reliable operation of the inspection robot.	1. RGB, the components of relevance is too large. 2. Each channel into the luminance information, easily affected by the surrounding environment. 3. The color of the human cognitive process is not matching, is not suitable for analysis and segmentation of color image.

AUTHOR	PROBLEM STATEMENT	METHODOLOGY	EVALUATION METRIC	RESULT	LIMITATION
Markov Decision Process-Based Resilience Enhancement for Distribution Systems: An Approximate Dynamic Programming Approach Chong Wang , Member, IEEE, Ping Ju , Senior Member, IEEE, Shunbo Lei , Member, IEEE 2020	Because failures in distribution systems caused by extreme weather events directly result in consumers' outages, this paper proposes a state-based decision-making model with the objective of mitigating loss of load to improve the distribution system resilience throughout the unfolding events.	A recursive optimization model based on Markov decision processes (MDP) is developed to make state-based actions. To overcome the curse of dimensionality caused by enormous states and actions, an approximate dynamic programming (ADP) approach based on post-decision states and iteration is used to solve the proposed MDP-based model.	Two test systems are used to verify the proposed model and the algorithm. The first system is the IEEE 33-bus system, and the second system is the IEEE 123- bus system. The cases are tested in MATLAB 2017a using the CPLEX 12.6 solver on computers with 3.1 GHz i5 processors and 8 GB RAMS	Case studies demonstrate that the state-based strategies try to make the feeders on the trajectory locate the terminal of the whole network to reduce potential loss of load, and in consequence to improve system resilience.	According to the decision processes, constructing the state transition tree under actions in consideration of uncertainties is one critical step to solve the MDP-based model. It is a difficult task to construct the state transition tree of the proposed model in consideration of various actions and the resulting complicated state transitions.

Use of Discrete and Continuous Markov Chains for System Absorbing States Tetiana Olekh Viktor Gogunskii 2018	The paper shows the use of Markov chains and oriented graphs in models of gradation of state of correspondence as the degree of project perfection. In the description of these models, the decomposition of the systems under investigation into certain discrete states is performed and a scheme of transitions between these states is created.	A fundamental matrix for a hypothetically simulated Markov absorption chain is considered, which gives the same prediction for the future regardless of the absolute value of the elapsed time.	A model for assessing the degree of compliance of environmental assessments with quality criteria is presented in the form of an oriented graph. The vertices of the graph correspond to the states of degrees of conformity of ecological estimates to certain criteria, and the arcs to nonzero transition probabilities.	The fundamental matrix gives the same prediction for the future, regardless of the absolute value of the time elapsed from the initial moment. This property of the fundamental matrix illustrates the Markov property of the process, characterizing it as a process without aftereffect	The nature of the probability distribution of the initial states is determined by the conditions of the problem. For example, at an initial moment the system may be in each of the states with equal probability. The appearance of the transition matrix depends entirely on numbering the states.

TECHNICAL SPECIFICATIONS

The primary benefits of parallel computing are the time saving and the possibility of better results. For example, due to the high computing efficiency, simulation models can be developed with more details for finite element analysis (FEA) and more search iterations can be used for optimisation studies, with the outcome that better simulation results and optimisation results can be achieved.

Parallel computing helps us harness power and make the most out of it. Parallel computing uses multiple computing units to perform multiple tasks simultaneously thus reducing the amount of time taken to perform a task stored in multiple computers with no way or a time consuming way to access them. Therefore we can see that parallel computing is emerging day to day to simplify the tasks and to access information faster and efficiently.

The algorithm demands parallel computing software. On using apriori and markov model we will be able to achieve the required goal of the project. The higher level implementation of the project uses sensors on the railway tracks that collects information about the train details and updates it on the software. This gives the user an idea about how and when an error might occur.

DESIGN

There has been an ever increasing rate of the number of trains being deployed and it has become very hard for us to keep up with this rise in the number of trains and the information regarding these trains. There were many methodologies which were adopted initially to make this process of monitoring trains easier like using sensors or supercomputers alone but later it was realised that using cloud computing using parallel distributed networks was more optimistic compared to the other methods.

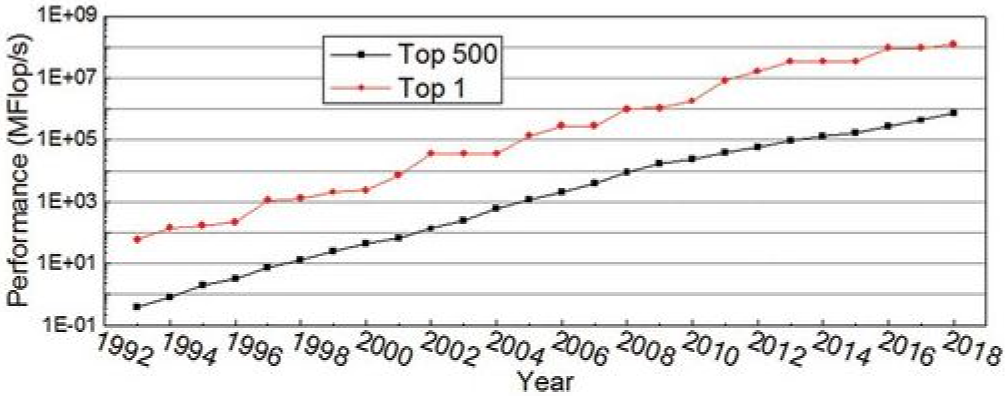


Figure 1. Supercomputing performance trends, data source, where MFlop/s means million floating point operations per second.

The above diagram shows the increase in the performance of monitoring trains over the years and it is to be duly noted that the increase in the performance was mainly due to the increase in parallel distributed networks.

This method uses K-Means algorithm to form clusters and based on these clusters formed we can monitor a train easily. The working of this algorithm in brief is given below

Step 1: Initialization

The first thing that K-Means does is randomly choose examples from the given dataset as initial centroids and that's simply it does not know yet where the center of each cluster is.

Step 2: Cluster Assignment

Then, all the data points that are closest (similar) to a centroid will create a cluster. If we're using the Euclidean distance between data points and every centroid, a straight line is drawn between two centroids, then a perpendicular bisector divides this line into two clusters.

Step 3: Move the centroid

Now, we have new clusters, that need centers. A centroid's new value is going to be the mean of all the examples in a cluster.

We'll keep repeating step 2 and 3 until the centroids stop moving, in other words, K-means algorithm is converged.

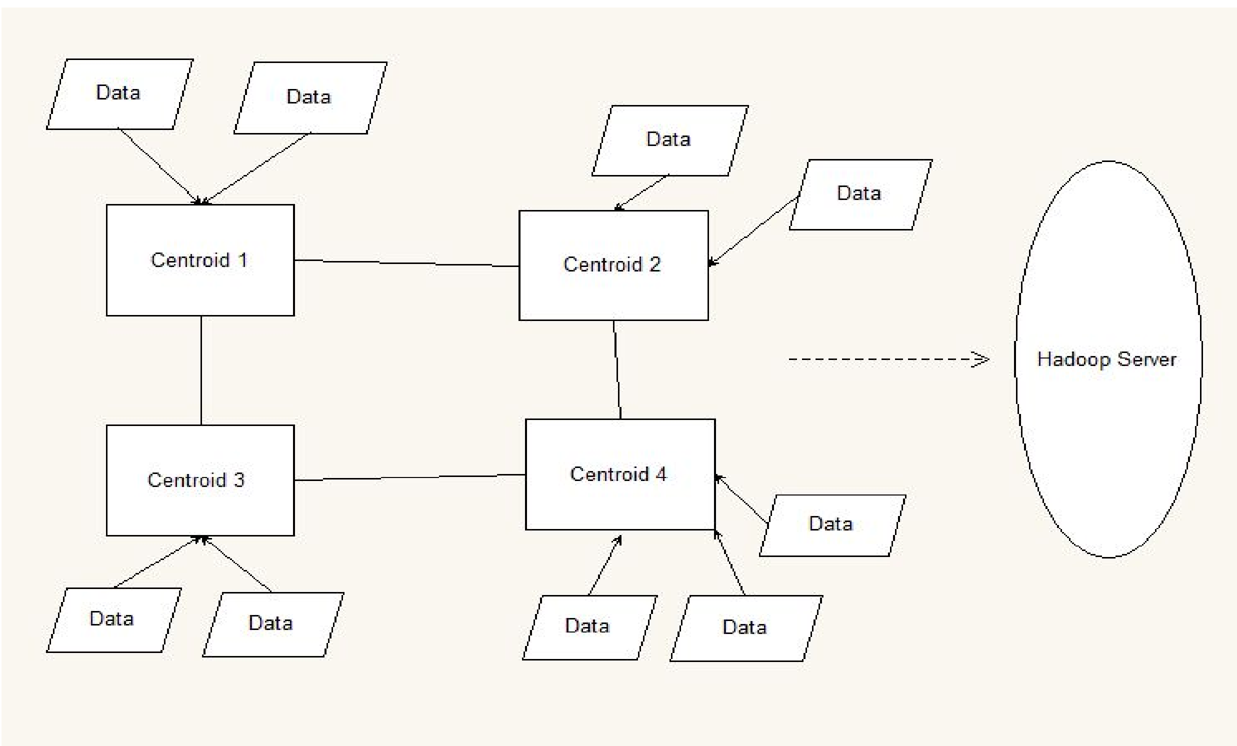


FIG 2: Diagrammatic representation of K-Means algorithm

PROPOSED SYSTEM :

MODULES :

Input Module :

This input module will consist of all the components that help pick up data from the field and supply it to the system for processing. These components can be anything like sensors on the railway tracks, computers accessed by the employees of the railway department, devices placed to check certain aspects like power and energy consumption, a passerby witnessing any mishap or even the cloud where all the data is finally stored.

This data devices are large in number and collect a large amount of data on a daily basis. It becomes really difficult to manage all of this data and classify it. For that purpose, we are using all these techniques of parallel computing for optimal processing.

Data and Signal Processing Module :

This large number of data collected by our input module needs to be processed and sorted out into useful and discardable information. For a long period of time this was being done manually and later with the use of a single system. Such large amounts of data will not even fit into a single system at any place. What we can do is store this data on the cloud and use applications like Hadoop.

In this module, we apply the MapReduce algorithm and the K-Means algorithm. The MapReduce algorithm helps us chunk out large pieces of data. It helps us take in and process that raw data into useful information. We can also use it to segregate data and see if it is useful. The K-Means algorithm is used to segregate this processed data into different clusters. We can have n number of clusters depending on the need of the system. This is used to make a searching process easier.

The outcomes of using this module is that we can process passenger flow traffic, process high speed train noise data, signal fault diagnosis, railway equipment management and ground penetrating radar data.

Activity Forecast Module :

In this module we implement the Markov model which is based on the Markov property. The Markov property states that the future state of a system depends only on the current state and not previous states, for a randomly changing system. A railway system is not entirely predictable. We have train schedules but still they run late, get cancelled or accidents occur. This uncertainty of the system is what makes us assume that it is a constantly changing one. This module is used to do the scheduling and forecast the possibility of accidents. The predictability of this system is crucial and needs to be optimal for the perfect functioning of railway systems.

Accident Factor Analysis :

Due to unforeseen circumstances, accidents occur. These accidents can lead to loss of life and damage to infrastructure and railway equipment. The railways would do anything in their power to prevent these mishaps. To do so, we need to analyse previous accidents and study them to understand what went wrong and then fix it.

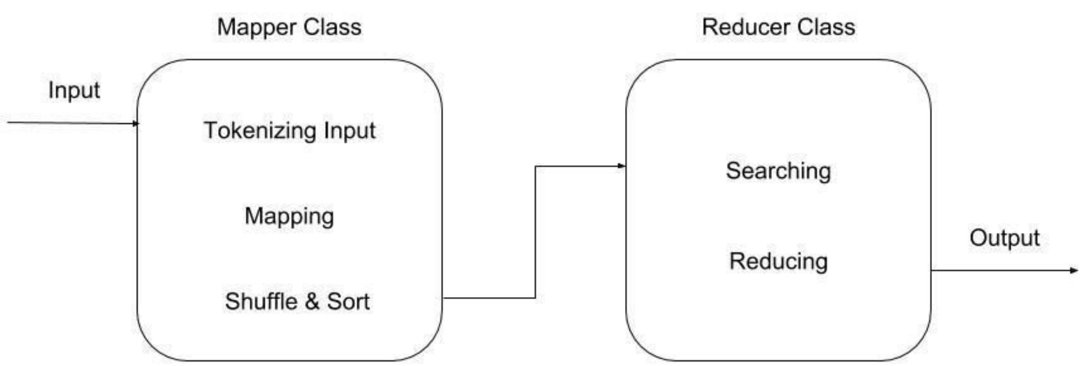
The best tool we can use in analysing large amounts of data is the Apriori algorithm. This algorithm helps us find patterns and previous occurrences in the dataset provided to us. It helps us generate a list of factors and reasoning behind the occurrence of these accidents and so we can use it to make changes to the system to prevent future accidents and irreparable damages.

ALGORITHMS :

TheMapReduce algorithm contains two important tasks, namely Map and Reduce.

- The map task is done by means of Mapper Class
- The reduce task is done by means of Reducer Class.

Mapper class takes the input, tokenizes it, maps and sorts it. The output of Mapper class is used as input by Reducer class, which in turn searches matching pairs and reduces them.



MapReduce implements various mathematical algorithms to divide a task into small parts and assign them to multiplesystems. In technical terms, MapReduce algorithm helps in sending the Map & Reduce tasks to appropriate servers in a cluster.

These mathematical algorithms may include the following −

•Sorting

Sorting is one of the basic MapReduce algorithms to process and analyse data. MapReduce implements sorting algorithm to automatically sort the output key-value pairs from themapper by their keys.

•Searching

Searching plays an important role in MapReduce algorithm. It helps in the combined phase (optional) and in theReducer phase. Let us try to understand how Searching works with the help of anexample.

•Indexing

Normally indexing is used to point to a particular data and its address. It performs batch indexing on the input files for a particular Mapper.

Theindexing technique that is normally used in MapReduce is known as invertedindex. Search engines likeGoogle and Bing useinvertedindexing technique. Let us try to understand how Indexing works with the help of a simpleexample.

•TF-IDF

TF-IDF is a text processing algorithm which is short for Term Frequency − Inverse Document Frequency. It is one of the common web analysis algorithms. Here, the term 'frequency' refers to the number of times a term appears in a document.

MAPPER CLASS

TheMapperclass defines the Map job. Maps input key-value pairs to a set of intermediate key-value pairs. Maps are the individual tasks that transform the input records into intermediate records. Thetransformedintermediate records need not be of the same type as the input records. A given input pair may map to zero or many output pairs.

REDUCER CLASS

TheReducerclass defines the Reduce job in MapReduce. It reduces a set of intermediate values that share a key to a smaller set of values. Reducerimplementations can access the Configuration for a job via theJobContext.getConfiguration() method. AReducer has three primary phases − Shuffle, Sort, and Reduce.

•**Shuffle** − TheReducercopies the sorted output from each Mapper using HTTP across thenetwork.

•**Sort** − The framework merge-sorts theReducer inputs by keys (since different Mappers may have output the same key). Theshuffle and sort phases occur simultaneously, i.e., while outputs are being fetched, they are merged.

•**Reduce** − In this phase the reduce (Object, Iterable, Context) method is called for each <key, (collection of values)> in the sorted inputs.

Generally MapReduce paradigm is based on sending map-reduce programs to computers where the actual data resides.

•During a MapReduce job, Hadoop sends Map and Reduce tasks to appropriate servers in the cluster.

•The framework manages all the details of data-passing like issuing tasks, verifying task completion, and copying data around the cluster between the nodes.

•Most of the computing takes place on the nodes with data on local disks that reduces thenetwork traffic.

•After completing a given task, the cluster collects and reduces the data to form an appropriateresult, and sends it back to the Hadoop server.

K-Means

Randomly choose k examples as centroids Whiletrue:

Create k clusters by assigning each example to closest centroid Compute k new centroids by averagingexamples in each cluster If centroids don’t change:

Break

K-means is a fast and efficient method, because thecomplexity of one iteration is k*n*d where k (number of clusters), n (number of examples), and d (time of computing theEuclidiandistance between 2 points).

We try different values of k, we evaluate them and we choose the best k value using the following algorithm:

Best = kMeans(points); For t in range(numTrials):

C= kMeans(points);

if dissimilarity(C) < dissimilarity(best): best = C;

return best

Dissimilarity(C) is the sum of all thevariabilities of k clusters

Variability is the sum of all Euclideandistances between the centroid and each example

in the cluster.

We have used Eucledian distance to calculate thedistance between every node and the centroid of every cluster to segregate the data points into different clusters. Eucledian distance function can be defined as:

Where x and y are two vectors

MARKOV

A Markov chain is a random process with the Markov property. A random process or often called stochastic property is a mathematical object defined as a collection of random variables. A Markov chain has either discrete state space (set of possible values of the random variables) or discrete index set (often representing time) - given the fact, many variations for a Markov chain exists. Usually the term "Markov chain" is reserved for a process with a discrete set of times, that is a Discrete Time Markov chain (DTMC).

A Markov chain is represented using a probabilistic automaton (It only sounds complicated!). The changes of state of the system are called transitions. The probabilities associated with various state changes are called transition probabilities. A probabilistic automaton includes the probability of a given transition into the transition function, turning it into a transition matrix.

If the Markov chain has N possible states, the matrix will be an N x N matrix, such that entry (I, J) is the probability of transitioning from state I to state J. Additionally, the transition matrix must be a stochastic matrix, a matrix whose entries in each row must add up to exactly 1. Why? Since each row represents its own probability distribution.

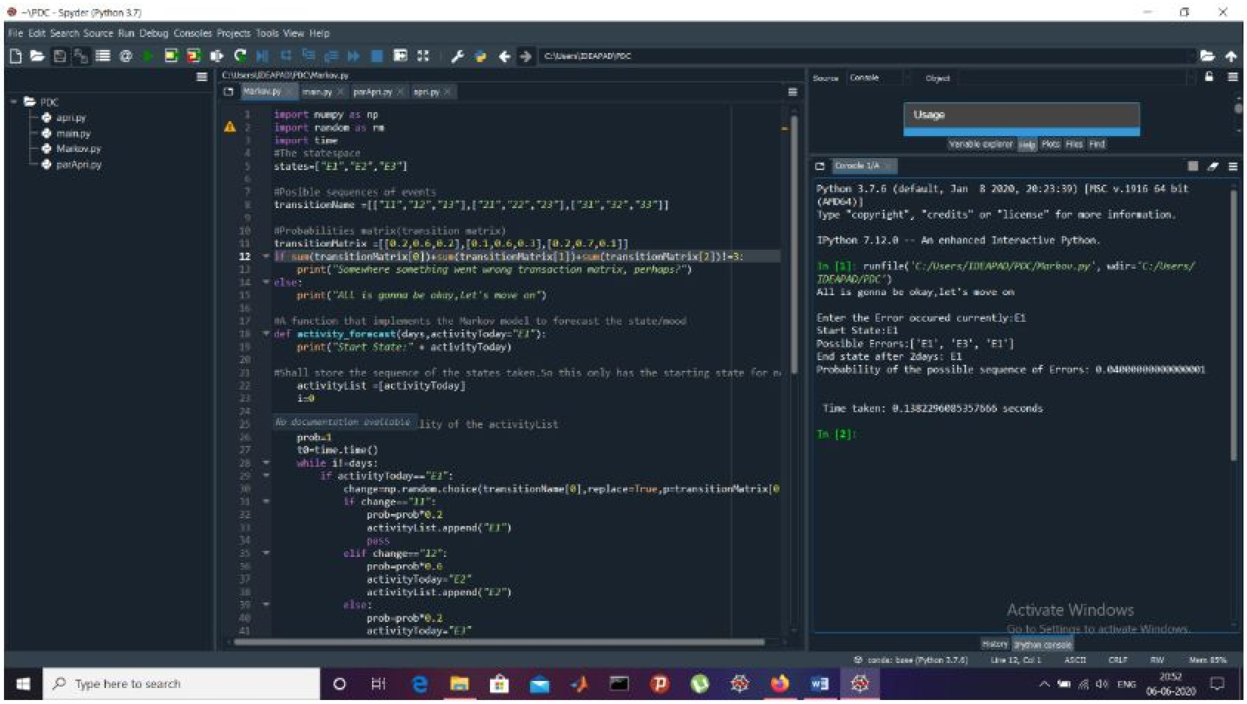
So, the model is characterized by a state space, a transition matrix describing the probabilities of particular transitions, and an initial state across the state space, given in the initial distribution.

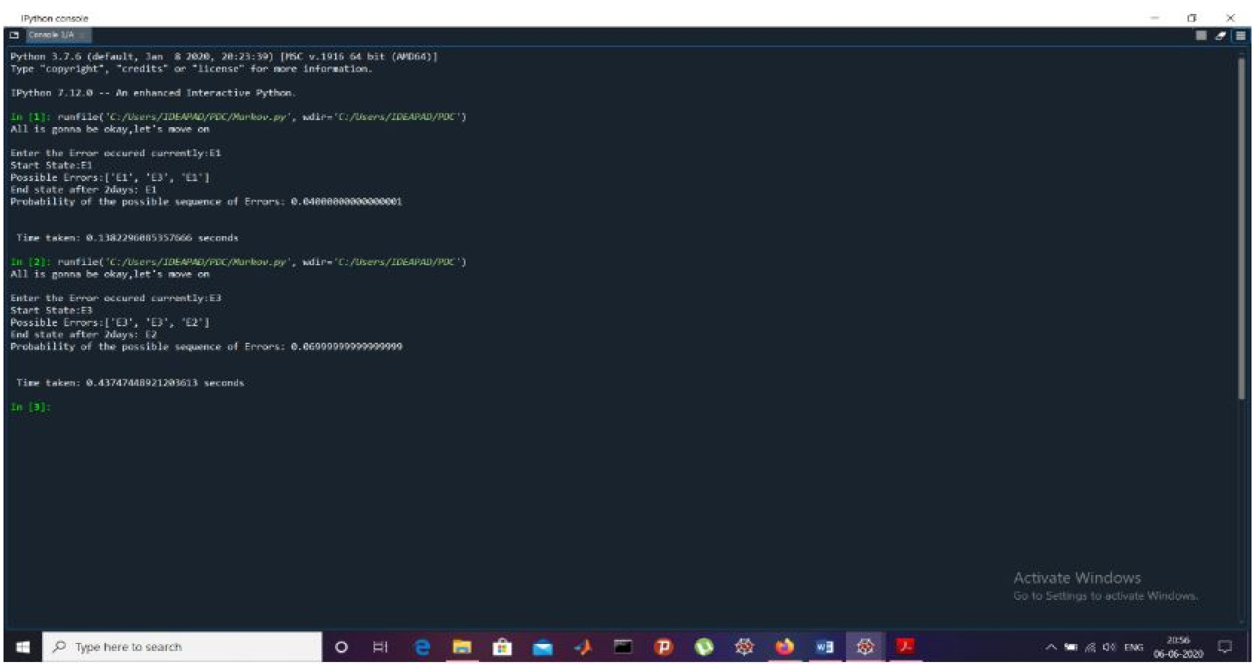
The Rules is a sequence of pair of strings, usually presented in the form of pattern → replacement. Each rule may be either ordinary or terminating.
Given an input string:

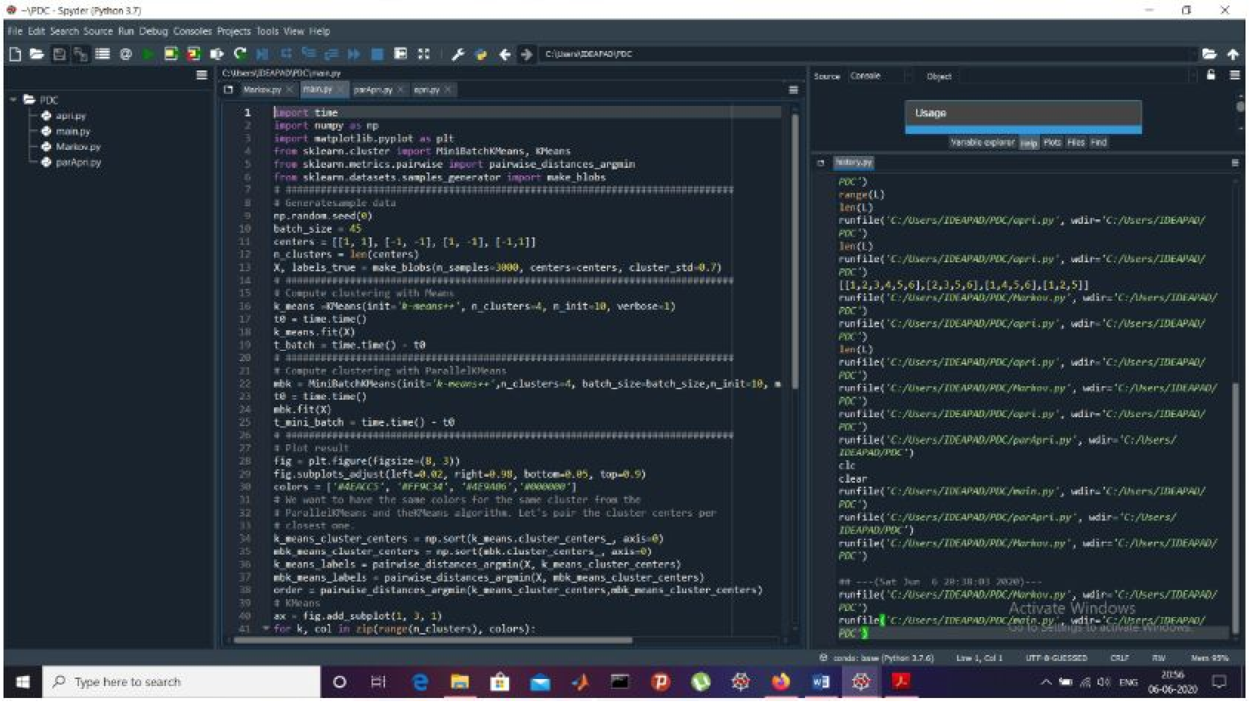
Check the Rules in order from top to bottom to see whether any of the patterns can be found in the input string.
If none is found, the algorithm stops.

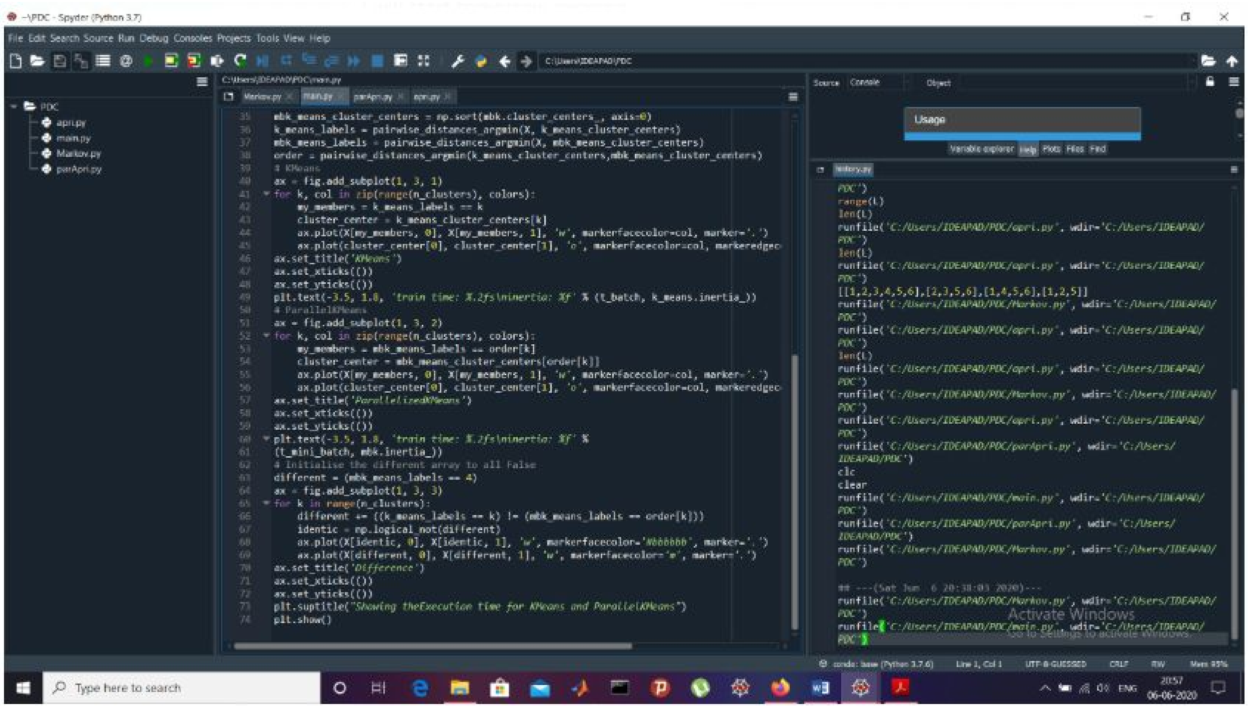
If one (or more) is found, use the first of them to replace the leftmost occurrence of matched text in the input string with its replacement.
If the rule just applied was a terminating one, the algorithm stops. Go to step 1.
Note that after each rule application the search starts over from the first rule.

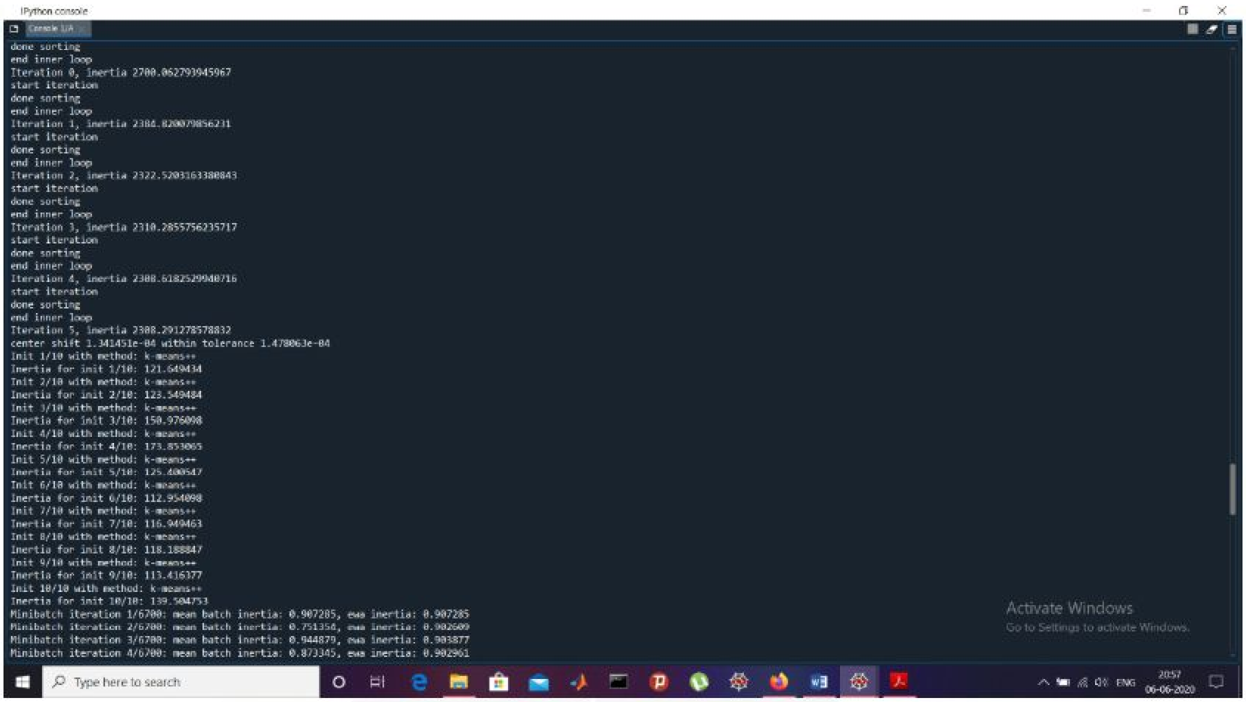
RESULTS AND DISCUSSION

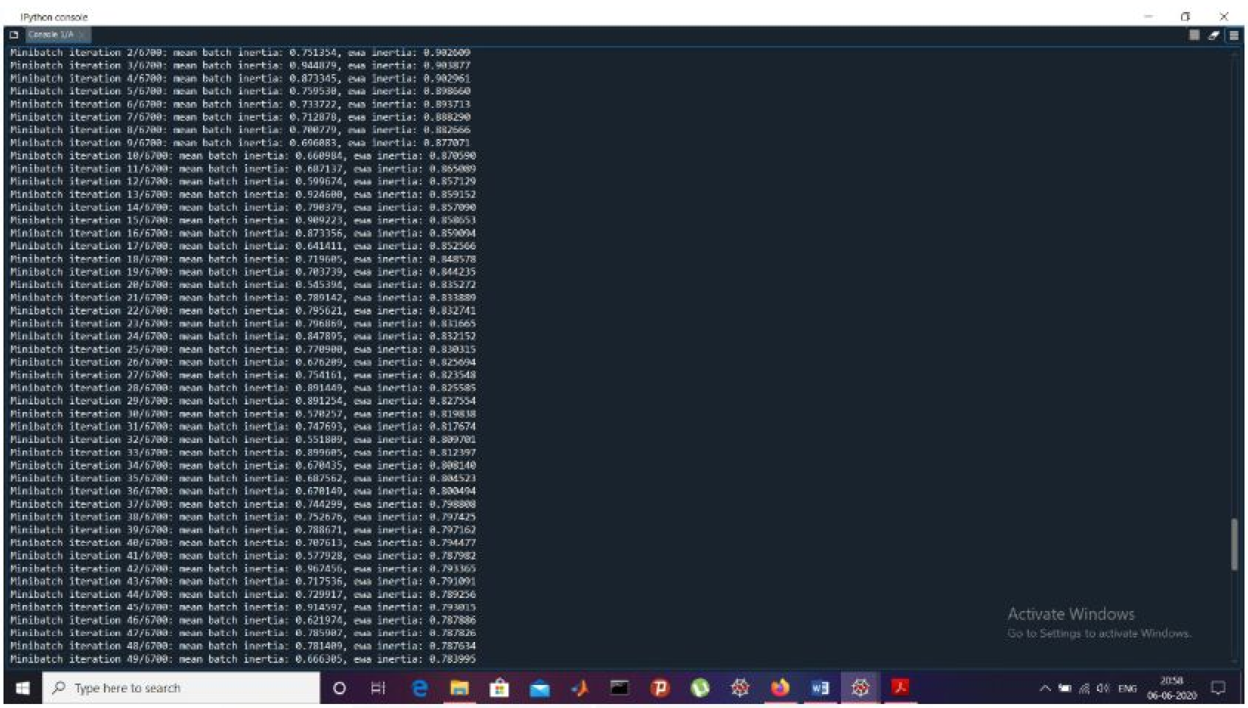




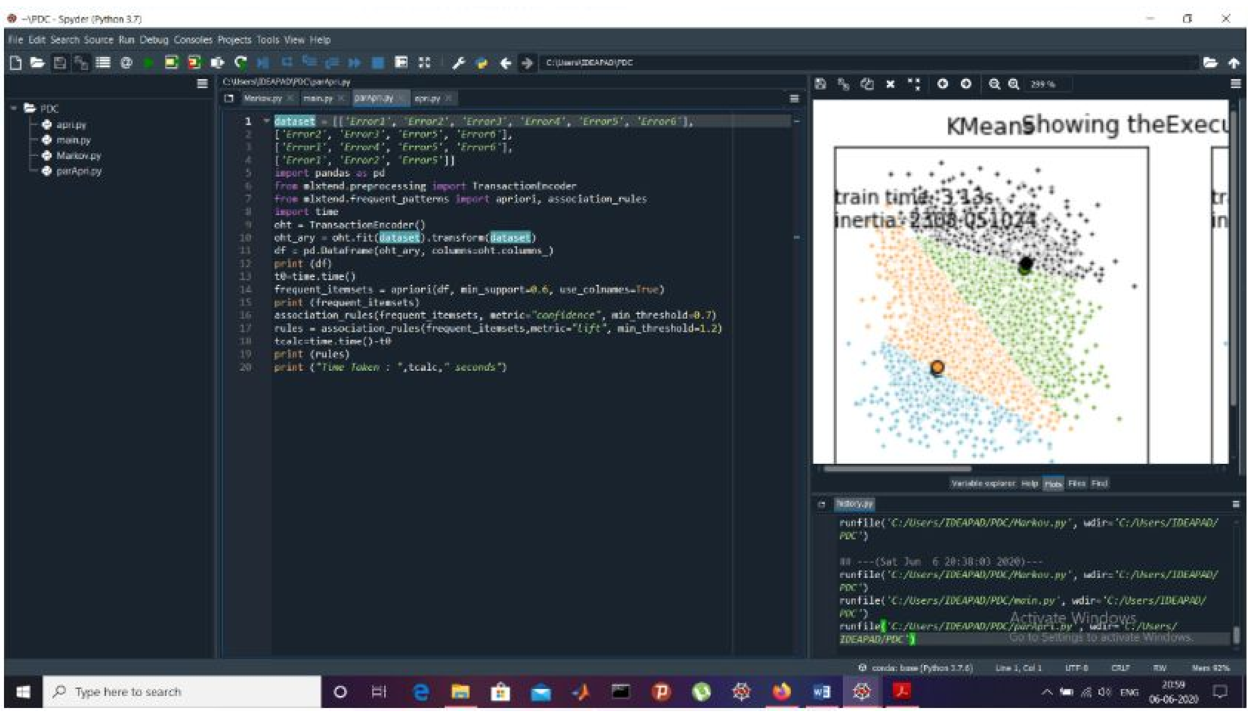

~\PDC - Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
import time
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import MiniBatchKMeans, KMeans
from sklearn.metrics.pairwise import pairwise_distances_argmin
from sklearn.datasets.samples_generator import make_blobs
# Generatesample data
np.random.seed(0)
batch_size = 45
centers = [[1, 1], [-1, -1], [1, -1], [-1,1]]
n_clusters = len(centers)
X, labels_true = make_blobs(n_samples=3000, centers=centers, cluster_std=0.7)
# Compute clustering with Means
k_means =KMeans(init='k-means++', n_clusters=4, n_init=10, verbose=1)
t0 = time.time()
k_means.fit(X)
t_batch = time.time() - t0
# Compute clustering with ParallelKMeans
mbk = MiniBatchKMeans(init='k-means++',n_clusters=4, batch_size=batch_size,n_init=10,
t0 = time.time()
mbk.fit(X)
t_mini_batch = time.time() - t0
# Plot result
fig = plt.figure(figsize=(8, 3))
fig.subplots_adjust(left=0.02, right=0.98, bottom=0.05, top=0.9)
colors = ['#4EACC5', '#FF9C34', '#4E9A06','#000000']
# We want to have the same colors for the same cluster from the
# ParallelKMeans and theKMeans algorithm. Let's pair the cluster centers per
# closest one.
k_means_cluster_centers = np.sort(k_means.cluster_centers_, axis=0)
mbk_means_cluster_centers = np.sort(mbk.cluster_centers_, axis=0)
k_means_labels = pairwise_distances_argmin(X, k_means_cluster_centers)
mbk_means_labels = pairwise_distances_argmin(X, mbk_means_cluster_centers)
order = pairwise_distances_argmin(k_means_cluster_centers,mbk_means_cluster_centers)
# KMeans
ax = fig.add_subplot(1, 3, 1)
for k, col in zip(range(n_clusters), colors):
Type here to search

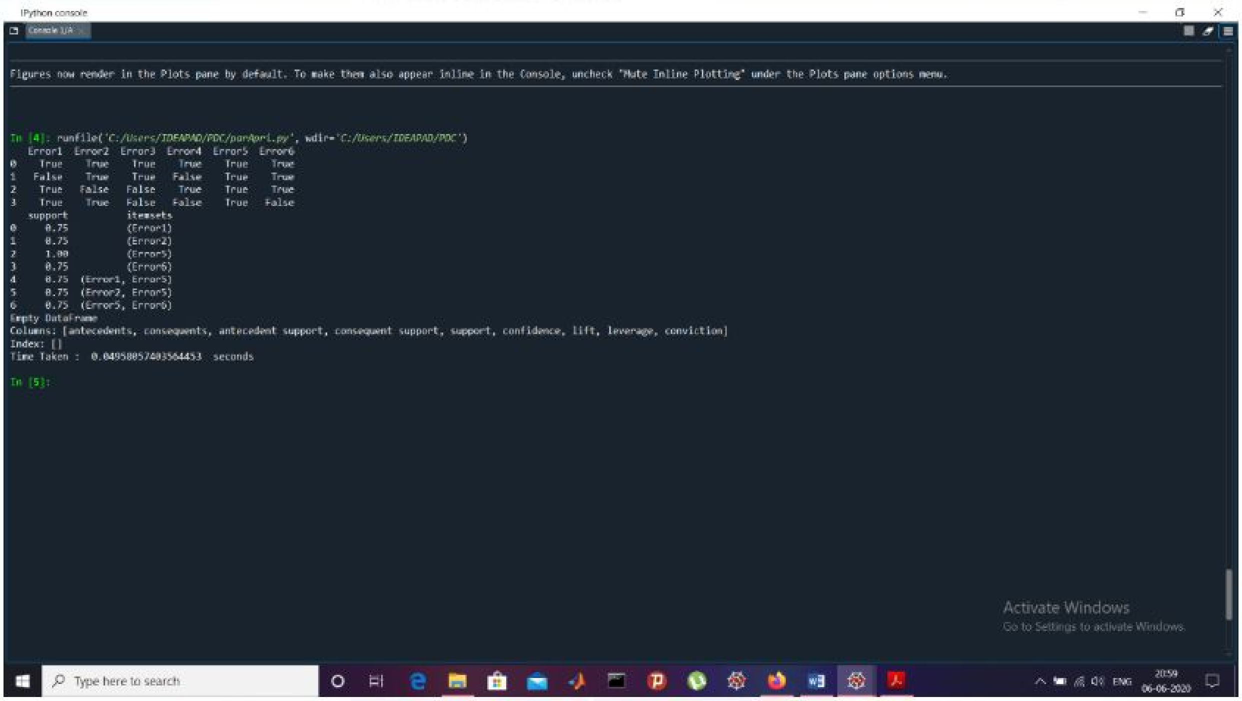

~\PDC - Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
mbk_means_cluster_centers = np.sort(mbk.cluster_centers_, axis=0)
k_means_labels = pairwise_distances_argmin(X, k_means_cluster_centers)
mbk_means_labels = pairwise_distances_argmin(X, mbk_means_cluster_centers)
order = pairwise_distances_argmin(k_means_cluster_centers,mbk_means_cluster_centers)
# KMeans
ax = fig.add_subplot(1, 3, 1)
for k, col in zip(range(n_clusters), colors):
my_members = k_means_labels == k
cluster_center = k_means_cluster_centers[k]
ax.plot(X[my_members, 0], X[my_members, 1], 'w', markerfacecolor=col, marker='.')
ax.plot(cluster_center[0], cluster_center[1], 'o', markerfacecolor=col, markeredgec
ax.set_title('KMeans')
ax.set_xticks(())
ax.set_yticks(())
plt.text(-3.5, 1.8, 'train time: %.2fs\ninertia: %f' % (t_batch, k_means.inertia_))
# ParallelKMeans
ax = fig.add_subplot(1, 3, 2)
for k, col in zip(range(n_clusters), colors):
my_members = mbk_means_labels == order[k]
cluster_center = mbk_means_cluster_centers[order[k]]
ax.plot(X[my_members, 0], X[my_members, 1], 'w', markerfacecolor=col, marker='.')
ax.plot(cluster_center[0], cluster_center[1], 'o', markerfacecolor=col, markeredgec
ax.set_title('ParallelizedKMeans')
ax.set_xticks(())
ax.set_yticks(())
plt.text(-3.5, 1.8, 'train time: %.2fs\ninertia: %f' %
(t_mini_batch, mbk.inertia_))
# Initialise the different array to all False
different = (mbk_means_labels == 4)
ax = fig.add_subplot(1, 3, 3)
for k in range(n_clusters):
different += ((k_means_labels == k) != (mbk_means_labels == order[k]))
identic = np.logical_not(different)
ax.plot(X[identic, 0], X[identic, 1], 'w', markerfacecolor='#bbbbbb', marker='.')
ax.plot(X[different, 0], X[different, 1], 'w', markerfacecolor='m', marker='.')
ax.set_title('Difference')
ax.set_xticks(())
ax.set_yticks(())
plt.suptitle("Showing theExecution time for KMeans and ParallelKMeans")
plt.show()
Type here to search

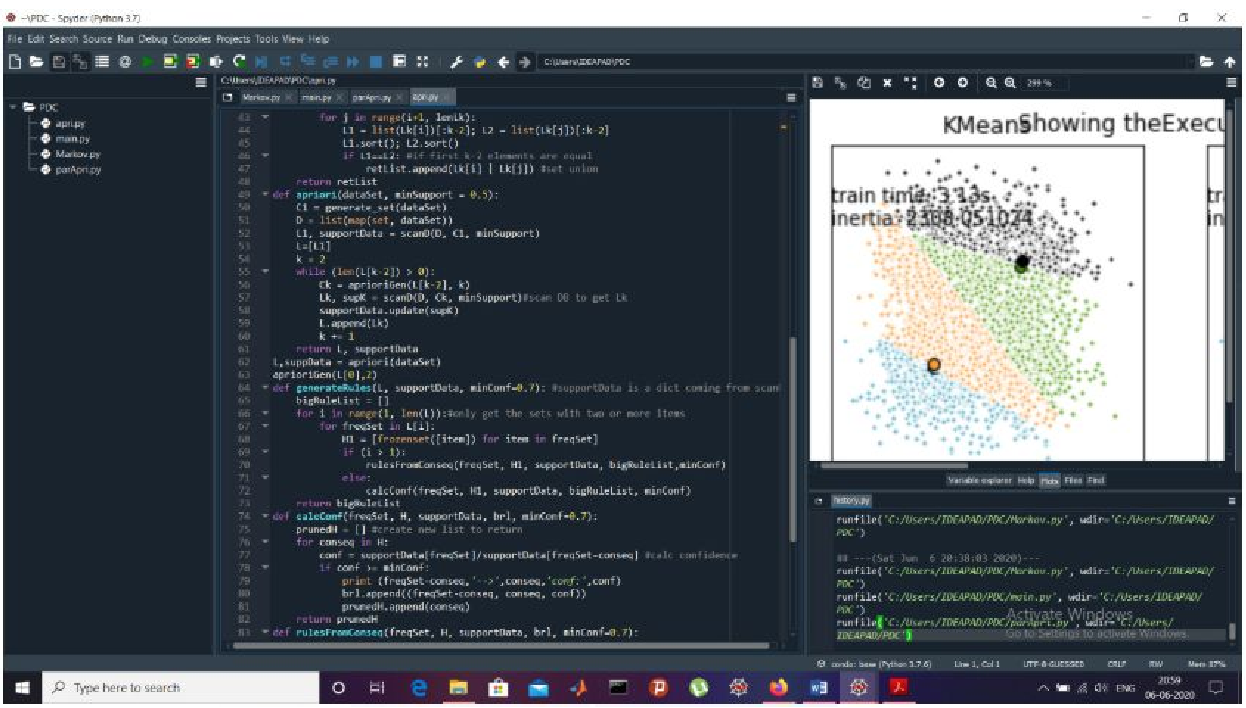

IPython console
Console 1/A
done sorting
end inner loop
Iteration 0, inertia 2700.062793945967
start iteration
done sorting
end inner loop
Iteration 1, inertia 2384.820079856231
start iteration
done sorting
end inner loop
Iteration 2, inertia 2322.5203163380843
start iteration
done sorting
end inner loop
Iteration 3, inertia 2310.2855756235717
start iteration
done sorting
end inner loop
Iteration 4, inertia 2308.6182529940716
start iteration
done sorting
end inner loop
Iteration 5, inertia 2308.291278578832
center shift 1.341451e-04 within tolerance 1.478063e-04
Init 1/10 with method: k-means++
Inertia for init 1/10: 121.649434
Init 2/10 with method: k-means++
Inertia for init 2/10: 123.549484
Init 3/10 with method: k-means++
Inertia for init 3/10: 150.976098
Init 4/10 with method: k-means++
Inertia for init 4/10: 173.853065
Init 5/10 with method: k-means++
Inertia for init 5/10: 125.400547
Init 6/10 with method: k-means++
Inertia for init 6/10: 112.954098
Init 7/10 with method: k-means++
Inertia for init 7/10: 116.949463
Init 8/10 with method: k-means++
Inertia for init 8/10: 118.188847
Init 9/10 with method: k-means++
Inertia for init 9/10: 113.416377
Init 10/10 with method: k-means++
Inertia for init 10/10: 139.504753
Minibatch iteration 1/6700: mean batch inertia: 0.907285, ewa inertia: 0.907285
Minibatch iteration 2/6700: mean batch inertia: 0.751354, ewa inertia: 0.902609
Minibatch iteration 3/6700: mean batch inertia: 0.944879, ewa inertia: 0.903877
Minibatch iteration 4/6700: mean batch inertia: 0.873345, ewa inertia: 0.902961
Activate Windows
Go to Settings to activate Windows.
Type here to search
ENG
20:57
06-06-2020


IPython console
Console 1/A
Minibatch iteration 2/6700: mean batch inertia: 0.751354, ewa inertia: 0.902609
Minibatch iteration 3/6700: mean batch inertia: 0.944879, ewa inertia: 0.903877
Minibatch iteration 4/6700: mean batch inertia: 0.873345, ewa inertia: 0.902961
Minibatch iteration 5/6700: mean batch inertia: 0.759530, ewa inertia: 0.898660
Minibatch iteration 6/6700: mean batch inertia: 0.733722, ewa inertia: 0.893713
Minibatch iteration 7/6700: mean batch inertia: 0.712870, ewa inertia: 0.888290
Minibatch iteration 8/6700: mean batch inertia: 0.700779, ewa inertia: 0.882666
Minibatch iteration 9/6700: mean batch inertia: 0.696083, ewa inertia: 0.877071
Minibatch iteration 10/6700: mean batch inertia: 0.660984, ewa inertia: 0.870590
Minibatch iteration 11/6700: mean batch inertia: 0.687137, ewa inertia: 0.865089
Minibatch iteration 12/6700: mean batch inertia: 0.599674, ewa inertia: 0.857129
Minibatch iteration 13/6700: mean batch inertia: 0.924600, ewa inertia: 0.859152
Minibatch iteration 14/6700: mean batch inertia: 0.790379, ewa inertia: 0.857090
Minibatch iteration 15/6700: mean batch inertia: 0.909223, ewa inertia: 0.858653
Minibatch iteration 16/6700: mean batch inertia: 0.873356, ewa inertia: 0.859094
Minibatch iteration 17/6700: mean batch inertia: 0.641411, ewa inertia: 0.852566
Minibatch iteration 18/6700: mean batch inertia: 0.719605, ewa inertia: 0.848578
Minibatch iteration 19/6700: mean batch inertia: 0.703739, ewa inertia: 0.844235
Minibatch iteration 20/6700: mean batch inertia: 0.545398, ewa inertia: 0.835272
Minibatch iteration 21/6700: mean batch inertia: 0.789142, ewa inertia: 0.833889
Minibatch iteration 22/6700: mean batch inertia: 0.795621, ewa inertia: 0.832741
Minibatch iteration 23/6700: mean batch inertia: 0.796869, ewa inertia: 0.831665
Minibatch iteration 24/6700: mean batch inertia: 0.847895, ewa inertia: 0.832152
Minibatch iteration 25/6700: mean batch inertia: 0.770900, ewa inertia: 0.830315
Minibatch iteration 26/6700: mean batch inertia: 0.676209, ewa inertia: 0.825694
Minibatch iteration 27/6700: mean batch inertia: 0.754161, ewa inertia: 0.823548
Minibatch iteration 28/6700: mean batch inertia: 0.891449, ewa inertia: 0.825585
Minibatch iteration 29/6700: mean batch inertia: 0.891254, ewa inertia: 0.827554
Minibatch iteration 30/6700: mean batch inertia: 0.570257, ewa inertia: 0.819838
Minibatch iteration 31/6700: mean batch inertia: 0.747693, ewa inertia: 0.817674
Minibatch iteration 32/6700: mean batch inertia: 0.551889, ewa inertia: 0.809701
Minibatch iteration 33/6700: mean batch inertia: 0.899605, ewa inertia: 0.812397
Minibatch iteration 34/6700: mean batch inertia: 0.670435, ewa inertia: 0.808140
Minibatch iteration 35/6700: mean batch inertia: 0.687562, ewa inertia: 0.804523
Minibatch iteration 36/6700: mean batch inertia: 0.670149, ewa inertia: 0.800494
Minibatch iteration 37/6700: mean batch inertia: 0.744299, ewa inertia: 0.798808
Minibatch iteration 38/6700: mean batch inertia: 0.752676, ewa inertia: 0.797425
Minibatch iteration 39/6700: mean batch inertia: 0.788671, ewa inertia: 0.797162
Minibatch iteration 40/6700: mean batch inertia: 0.707613, ewa inertia: 0.794477
Minibatch iteration 41/6700: mean batch inertia: 0.577928, ewa inertia: 0.787982
Minibatch iteration 42/6700: mean batch inertia: 0.967456, ewa inertia: 0.793365
Minibatch iteration 43/6700: mean batch inertia: 0.717536, ewa inertia: 0.791091
Minibatch iteration 44/6700: mean batch inertia: 0.729917, ewa inertia: 0.789256
Minibatch iteration 45/6700: mean batch inertia: 0.914597, ewa inertia: 0.793015
Minibatch iteration 46/6700: mean batch inertia: 0.621974, ewa inertia: 0.787886
Minibatch iteration 47/6700: mean batch inertia: 0.785987, ewa inertia: 0.787826
Minibatch iteration 48/6700: mean batch inertia: 0.781489, ewa inertia: 0.787634
Minibatch iteration 49/6700: mean batch inertia: 0.666305, ewa inertia: 0.783995
Activate Windows
Go to Settings to activate Windows.
Type here to search
ENG
20:58
06-06-2020


KMeansShowing theExecutionParallelizedKMeans and ParallelKMeansDifference
train time: 3.13s
inertia: 2308.051024
train time: 0.26s
inertia: 2319.611645
Activate Windows
Go to Settings to activate Windows.
Type here to search


dataset = [['Error1', 'Error2', 'Error3', 'Error4', 'Error5', 'Error6'],
['Error2', 'Error3', 'Error5', 'Error6'],
['Error1', 'Error4', 'Error5', 'Error6'],
['Error1', 'Error2', 'Error5']]
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules
import time
oht = TransactionEncoder()
oht_ary = oht.fit(dataset).transform(dataset)
df = pd.DataFrame(oht_ary, columns=oht.columns_)
print (df)
t0=time.time()
frequent_itemsets = apriori(df, min_support=0.6, use_colnames=True)
print (frequent_itemsets)
association_rules(frequent_itemsets, metric="confidence", min_threshold=0.7)
rules = association_rules(frequent_itemsets,metric="lift", min_threshold=1.2)
tcalc=time.time()-t0
print (rules)
print ("Time Taken : ",tcalc," seconds")
KMeansShowing theExecu
train time: 3.13s
inertia: 2308.051024
Activate Windows
Type here to search


IPython console
Figures now render in the Plots pane by default. To make them also appear inline in the Console, uncheck "Mute Inline Plotting" under the Plots pane options menu.
In [4]: runfile('C:/Users/IDEAPAD/PDC/parApri.py', wdir='C:/Users/IDEAPAD/PDC')
   Error1 Error2 Error3 Error4 Error5 Error6
0   True   True   True   True   True   True
1   False   True   True   False   True   True
2   True   False   False   True   True   True
3   True   True   False   False   True   False
   support   itemsets
0   0.75   (Error1)
1   0.75   (Error2)
2   1.00   (Error5)
3   0.75   (Error6)
4   0.75 (Error1, Error5)
5   0.75 (Error2, Error5)
6   0.75 (Error5, Error6)
Empty DataFrame
Columns: [antecedents, consequents, antecedent support, consequent support, support, confidence, lift, leverage, conviction]
Index: []
Time Taken : 0.04958057403564453 seconds
In [5]:
Activate Windows
Go to Settings to activate Windows.
Type here to search


~\PDC - Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
Type here to search

CONCLUSION

The primary benefits of parallel computing are the time saving and the possibility of better results. For example, due to the high computing efficiency, simulation models can be developed with more details for finite element analysis (FEA) and more search iterations can be used for optimisation studies, with the outcome that better simulation results and optimisation results can be achieved.

Parallel computing helps us harness power and make the most out of it. Parallel computing uses multiple computing units to perform multiple tasks simultaneously thus reducing the amount of time taken to perform a task stored in multiple computers with no way or a time consuming way to access them. Therefore we can see that parallel computing is emerging day to day to simplify the tasks and to access information faster and efficiently.

We successfully implemented all the algorithms that we intended to and parallelised them as well. We hope to implement the project on a larger scale some day if given the opportunity to.

REFERENCES

1. L. M. Rodrigues, L. E. Zárate, C. N. Nobre and H. C. Freitas, "Parallel and distributed kmeans to identify the translation initiation site of proteins," *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Seoul, 2012, pp. 1639-1645, doi: 10.1109/ICSMC.2012.6377972.
2. T. Song, J. Li, X. Han and L. Shao, "A quick Otsu-Kmeans algorithm for the internal pipeline detection," *2017 IEEE International Conference on* Mechatronics and Automation (ICMA), Takamatsu, 2017, pp. 193-197, doi: 10.1109/ICMA.2017.8015812.
3. Martin L. Puterman. 1994. Markov Decision Processes: Discrete Stochastic Dynamic Programming (1st. ed.). John Wiley & Sons, Inc., USA.
4. T. Olekh and V. Gogunskii, "Use of Discrete and Continuous Markov Chains for System Absorbing States," 2019 IEEE International Conference on Advanced Trends in Information Theory (ATIT), Kyiv, Ukraine, 2019, pp. 518-521, doi: 10.1109/ATIT49449.2019.9030457.
5. Potuzak, Tomas. "Optimization of a genetic algorithm for road traffic network division using a distributed/parallel genetic algorithm." In 2016 9th International Conference on Human System Interactions (HSI), pp. 21-27. IEEE, 2016.
6. Yabing, J. (2013). Research of an improved apriori algorithm in data mining association rules. *International Journal of Computer and Communication Engineering*, *2*(1), 25.
7. Ajeesh, K. P., Ashis, R., Muttath, M. H., Chandran, N. K., Sreedevi, C. R., & Praveen, R. P. (2012, August). A novel fuzzy logic controller based

automatic caution order system for Indian railways. In *2012 IEEE International Conference on Advanced Communication Control and* Computing Technologies (ICACCCT) (pp. 248-253). IEEE.

8. Patel, Hiral M. "A comparative analysis of MapReduce scheduling algorithms for Hadoop." Int. J. Innov. Emerg. Res. Eng 2.2 (2015).
9. Wu, Q., Spiryagin, M., Cole, C., & McSweeney, T. (2018). Parallel computing in railway research. International Journal of Rail Transportation, 1-24.
10. Sharma, Apekshit, and Chandrakant Sharma. "DISTRIBUTED COMPUTING IN MOBILE ENVIRONMENT."