Performance Enhancement of Customer Segmentation Using a Distributed Python Framework, Ray

Debajit Datta, Rishav Agarwal, Preetha Evangeline David

Abstract— Over the years, there has been a huge popularity of the recommender systems worldwide. Recommender systems have been implemented over several domains ranging from recommendations for videos and movies to that for products and applications, and many more. The algorithms, which are used for recommender systems, implement segmentation of the customer based on several attributes. These algorithms are time-consuming and require comparatively high computation power. This work deals with the parallelization of different algorithms for simple customer segmentation in the Python environment using the framework, Ray. The dataset for this work includes a huge list of purchases that are carried out by 4000 customers, over a year. The parallelization is carried out throughout the multicores of CPU and the cores of GPU. Additionally, the work also shows the speedup that is obtained after parallelization, for analyzing the overall increase in performance.

Index Terms— Accuracy Metrics, Classification, Clustering, CPU, GPU, Parallel Computing, Recommendation, Segmentation, Speedup.

1 INTRODUCTION

SINCE the advent of customer-based online platforms, be it in terms of online streaming channels or the online shopping platforms, recommendation systems have played an important role as one of the most important features for the same. The current generation thrives on this recommendation feature and so do the current marketers. Knowing the purchase history of the customers, online shopping platforms gain an access to the probable products that the customer might like to buy. A customer buying a formal shirt might soon be buying a formal tie, a customer buying a pair of pants might soon be requiring a belt. All these transactions allow the marketers, companies, online shopping platforms to pitch those recommendations to the customers which they know will stick and compel to buy, thus, increasing their profits. Coming to the online streaming platform, if a user streams thriller movie, they would be recommended thriller movies, as those streaming platforms understand that the chances of that user persistently uses their platform is when he gets recommended new streaming options in the category they like. This proves that recommender systems are the crux of online marketing.

All these recommender systems work on various algorithms which use the concept of customer segmentation in order to get the appropriate output. Customer segmentation is basically the grouping of various customers into various clusters based on their preferences which allows the market to target them specifically with specific products which would cater to their own interests and compel them to buy the same. Customer segmentation involves the use of various attributes like the industry that is being catered, the location of the customer, and the purchase history of the customers. The recommender systems work mostly on purchase or streaming histories in order to segment customers and target them with the required content. The recommender systems work on various algorithms that uses single CPU computation power. Algorithms like the k-Nearest Neighbor clustering, the collaborative filtering, the content-based methods, and the hybrid method that uses collaborative and content methods both. A simple recommender system can be implemented using these algorithms. The Nearest Neighbor algorithm clusters various customers with the most similar attributes as one and recommends them the same content. The collaborative method works solely on the customer's history of transaction, thus, fails to account for initial recommendation [6][18]. The content-based method gathers information about the customer and then recommends the content accordingly. The hybrid which is most profoundly used employs both the collaborative and content model to recommend contents to its users. Using a simple python code, the recommender system can easily be implemented in a single-core CPU system. The users do not notice the delay that these algorithms cause when recommending the appropriate contents, but these inefficiency of the algorithms causes long-term losses to the online platforms and affects the customer satisfaction to a great extent. A customer should not be allowed to wait before appropriate recommendation start appearing while they are having the e-commerce experience

The process of parallelizing a program increases the efficiency of any algorithm to a great extent. It is the process where the computer breaks down the tasks given to it and works on it parallelly. This process is facilitated by multi-core CPUs and GPUs. When compared to CPU, GPU is known provide greater computation power with minimal overhead. Various frameworks and platforms can be used to implement parallel processing in GPUs [23].

The work presented by this paper aims to enhance the

- *Debajit Datta is currently pursuing bachelors degree program in computer science and engineering in Vellore Institute of Technology, Vellore, India. E-mail: debajit.datta2000@gmail.com*
- *Rishav Agarwal is currently pursuing bachelor's degree program in computer science and engineering in Vellore Institute of Technology, Vellore, India. E-mail: rishavagarwal2717@gmail.com*
- *Dr. Preetha Evangeline David is an assistant professor (Sr. Grade) of computer science and engineering (SCOPE) in Vellore Institute of Technology, Vellore, India. E-mail: preetha.evangeline@vit.ac.in*

experience of a recommendation system. The main motive is to enhance the customer segmentation algorithms by using the concepts of parallelization. The paper also presents a juxtaposition of the simple algorithm being parallelized through the multicores of a CPU and through the cores of a GPU [17][22]. The analysis for the same is done by calculating the speedup of the two parallelization processes. Using the concepts of Amdahl's law, the speedup which relates to the performance of the systems is calculated. In order to implement the said parallelization, the distributed framework Ray was used. It is very efficient when it comes to handling large data as it has shared-memory object [24]. The dataset used huge list of purchases that are carried out by 4000 customers, over a year. Seeing the size of the dataset, it is beneficial to use Ray as the framework as it can perform parallel computing at a much more efficient rate. The work brings out a much more efficient method to employ the customer segmentation algorithms in recommender systems. It brings out the major time efficiency difference when recommendations are expected. The final work not only provides us a way to better market the product to the target customers but also provide the customers a better experience as the current demographic thrives on the concept of recommender systems.

2 Related Work

According to the work of Kabasakal [1], Recency Frequency Monetary model has been suggested as an efficient method of customer segmentation. They have talk about the using customer segmentation to implement Customer Relationship Management. The case study has taken a three-dimensional approach using recency, frequency and the monetary factors of the customers. They have implemented the customer segmentation process successfully clustered customers based on similar attributes. They have successfully created a prototype which provides for the RFM model. According to the work by Maka Alkhayrat et. al. [2], talks about specifically about the customer base in a telecom market. It talks about customer segmentation and how unnecessary and redundant features can be removed, using dimensionality reduction, in order to get a better clustering of the customers based on more appropriate features. They use the Principal Component Analysis to reduce the dimensional data and then perform K-Means clustering on both the original and the reduced data, in order to bring out the comparison in both. They conclude that the customer segmentation done after dimensionality reduction produce better clustering and distribution of the customers.

According to paper by Bambang Eka Cahyana et. al. [3], the efficient method to carry out the process of customer segmentation is a hybrid technology. The work talks about clustering of sea transportation users using a hybrid algorithm. They combine hierarchical clustering and non-hierarchical clustering in order to gain greater flexibility and to remove the disadvantages and limitations caused by a single algorithm. The work of Matthias Carnein et. al. [4], talks about how customer segmentation needs to be carried out repetitively as the criteria for segmenting keeps changing. If the attribute is that of transaction history, then as the transaction's history updates, so should the customer segmented clusters. They have proposed the use of stream clustering which allows for a more scalable clustering process. The paper has taken a real-life scenario and taken a stream of input to implement the said algorithm and conclude its effectiveness. According to the work of Phan Duy Hong et. al. [5], using Hierarchical Agglomerative Clustering can be one of the methods to implement customer segmentation. The work uses R as the programming language and implements the said algorithm using a credit card dataset of the customers. They conclude that using the said approach appropriate clustering can be formed in order to segment the customers.

According to Datta et. al. [6], parallelization is an efficient method to perform any algorithm in a much more better time frame. The work does a comparative study of the parallel computation of CPU cores on a given a Convolutional Neural Network model. It showcases how accurate the model is when trained on a different number of CPU cores, while parallelizing the multicore CPU using Ray framework. It concludes that an image classification on a Convolutional Neural Network implemented on parallelized CPUs gives a very low computation time, thus, providing for an efficient approach. The work of Sertac Karahoda et. al. [7], talks about the process of synchronization in finite state machine. Generating the short sequences still turns out to be a NP-Hard problem, thus, they suggest the use of heuristics. In order to make it even faster and feasible, the algorithm is improved by parallelization of the aid heuristics. The paper compares the heuristic approach by sequential programming and parallel programming and concludes the efficiency of the latter. They show how the speedup value increases when parallelized CPU and GPUs are compared, favoring the latter. The paper by Daniel Gerzhoy et. al. [8], explores the heterogeneous microprocessors which integrate CPU-GPU chips allowing for a much better computation timeframe. A nested MIMD-SIMD parallelization is performed in order to compare the same in CPU- only and the one in the integrated CPU-GPU chips. The paper concludes that the latter is a more efficient method to carry out such implementations. The work of Sang Hee Kim et. al. [9], deals with the juxtaposition of CPU and GPU parallelization for a virtual heart simulation. It explains how a high computation power is required for a high-resolution mesh as it has numerous nodes and tissue movements. It performs the ordinary differential equation and partial difference equation in parallelized CPU and GPU. It concludes that the later increases the efficiency by a factor of 4.3 times and 2.3 times respectively.

According to the survey conducted by Mokhtar Essaid et. al. [10], parallel computation has been considered to solve metaheuristic hard problems. They have suggested the use of Graphic Processing Units-GPUs- in order to facilitate parallelization and have shown various survey results and research paper implementing the same. The paper by Duane Rosenberg et. al. [11], has suggested GPU parallelization using the CUDA framework. It has taken a hybrid MPI-OpenMP scheme and implemented it using CUDA to achieve

parallelization of GPU. It concluded with an increase in efficiency and speed-up of the said implementation. The work of Mayra Rodriguez [12], deals with a comparative approach of various clustering algorithms. The paper performs clustering using nine algorithms on the R-platform. They compare all the algorithms on a particular dataset and conclude on the most optimized algorithm to be used in a given scenario.

The paper by B. G. Mamatha Bai et. al. [13], deals with a Big Data Analytics approach to the health care system. It deals with using various clustering algorithms to segment the patients and produce results like the intensity of disease of diabetes and the medical assistance that they might require. They use different performance metrics to compare the result of the various algorithms. They conclude that different algorithms can be used based on the activity that needs to be performed during the analysis. The survey by Shuai Zhang et. al. [14], targets the various research works that have been done on a deep-learning based recommender system. They talk about Autoencoder Recommender systems, and Deep neural networks being implemented for recommendation in various online domains. They have successfully presented the advantages and disadvantages of using such recommender systems.

According to the work by Timur Osadchiy et. al. [15], Pairwise association rules can be used for a recommender system. It deals with clustering based on collective features or attributes which do not confer to the individual interests. They use a dataset of real-world dietary intake recall system. The work of Natarajan et. al. [16], deals with resolving cold-start problems in a collaborative filtering recommender system. They propose the use of Linked Open Data. They also focus on the problem of data sparsity that is encountered in recommender systems. The proposed methodology is the use of matrix factorization with open linked data- MF-LOD. They successfully resolve the problems faced by recommender systems using the said approach.

3 Proposed Work

Customer segmentation is the process by which the marketeers, be it the online markets or the offline markets, demarcate the different demographic of their customers. It is highly used in the concept of online recommender systems. The online platforms like streaming and shopping platforms use the concept of customer segmentation to provide with target advertisements to specific user, which would be most beneficial to the user and the company. Customer segmentation uses various data. Online streaming platforms sometimes ask the users their preference and then recommend the movies or television shows. The most common form of recommendation is done by using the history of usage of the customer. An online shopping platform would look into the customers' previous transaction then recommend products which would complement the ones they bought. A streaming platform would look at the viewers history and then recommend them shows which conform to that genre. Currently, the implementation of customer segmentation is carried out in the form of sequential programming, mostly using CPU.

In a digitized world as ours, speed, efficiency, productivity, cost levied by it, all these are the most essential aspect of solving any problems. The questions that are poised nowadays is not whether the problem can be solved, it is whether the problem can be solved in a more efficient manner. Sequential programming is the age-old method of solving all problems, which does not have any error and gives the required output [19][30], but that is not enough. The ever-evolving industries cannot survive with age-old methods that works just fine.

Sequential programming may still be used widely but it is not amiss of certain drawbacks which make it less efficient in the long run. Sequential programming is basically the method where the device performs a single instruction and moves on to the next one once the previous one is carried out. Thus, more the instructions that needs executing more is the overhead, more is the time taken to execute the same, and the overall cost increases. In case of customer segmentation, there are countless customers that need to be segmented into even more attributes. In case of streaming platforms, location, genre, search history come into play. In case of shopping platforms, the purchase history, age demographic, location come into play. When all these are evaluated sequentially, the overhead speed is a lot and the output, that is, the recommendations are either flawed or takes a lot of time. The way to overcome this is to parallelize the concept of customer segmentation and then implement the same in recommender systems [22][27].

The juxtaposition of various algorithms being implemented using multi-core CPUs and GPUs have always been implemented in order to showcase the widespread efficiency of the latter. In order to bring out this comparison in increasing the efficiency of customer segmentation, the work uses various algorithms and implements the same in CPU and GPU. The work employs various algorithms in sequential and parallelized programming by executing it in CPU with two, three and four cores. Then it uses GPU to perform the parallelization. The speedups for all the implementation are calculated and the best method to increase the efficiency of customer segmentation is concluded.

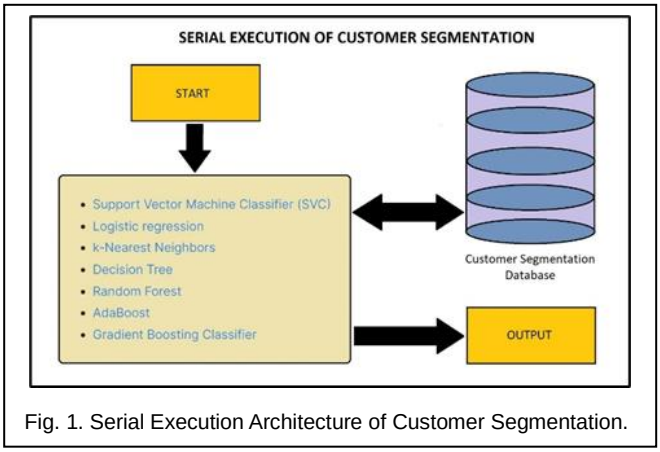

Fig. 1. Serial Execution Architecture of Customer Segmentation.

The Fig. 1 displays the architecture of a serial execution of

algorithms for customer segmentation. According to this, each algorithm goes through the execution sequentially taking longer time for the completion of the same. Efficiency reduces considerable as the overhead increases. The execution of a particular task does not take place until the previous task is completely executed.

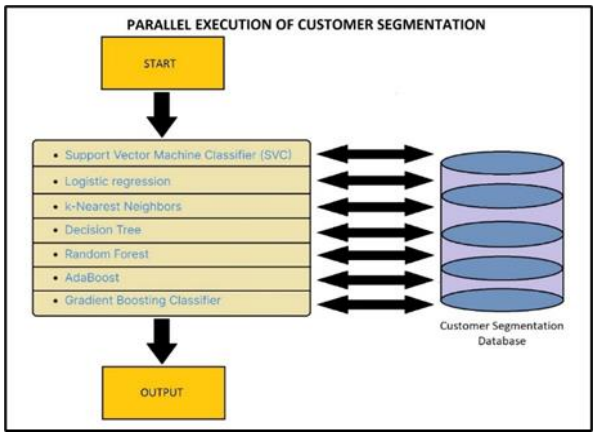

Fig. 2. Parallel Execution Architecture of Customer segmentation.

The Fig. 2 displays the architecture of a parallel execution of algorithms for customer segmentation. According to this, each algorithm can be simultaneously be executed, thus, improving the efficiency to a much greater extent, while ensuring that the time taken to come up with the output is minimal. Parallelization is basically the breaking down of an instruction into smaller instructions and using multiple processors to execute them at the same time. This increases the computational power while also decreasing the time taken in order to execute a single instruction. In order to parallelize a CPU, the use of multi-core CPU with 2 or more cores is done. Each core performs a certain task, as the instructions are divided as such. The execution of the tasks takes place simultaneously, thus, allowing for an efficient completion time. Python's module can be used to parallelize a multi-core CPU. The said module uses the concept of Pool distribution where the tasks are located in a pool from which it is allotted to various cores of the CPU. Graphic Processing Unit (GPU) allows for better shared memory which increases the computation power of the system. Parallelization of GPU can increase the efficiency of any program tremendously. The benefit of using GPU is that it consists of hundreds of small cores which allows parallel execution of multiple instructions. Thus, the work uses this to implement customer segmentation and see how much better is it as compared to parallelization in CPU.

In order to implement parallelization, many frameworks and platforms like CUDA, OpenCL and multiprocessing library can be used. But frameworks like python's library multiprocessing are not safe for multi-core processing, since they do not provide shared variable implementation and also are not friendly with large dataset. The work uses the framework Ray, that is used exclusively for applications in the field of artificial intelligence by parallelizing CPUs. It can also parallelize GPUs, and can detect the number of cores for both on its own. To use Ray, the Python library for the same can be used which allows for the implementation of both CPUs and GPUs parallelization. Once the implementation of all the algorithms are carried out using multi-core CPUs, the work, presented here, moves towards implementing the same algorithms using GPU.

Once the computation is completed, the speedup of both the processing units are calculated in order to compare and contrast between the two. Different algorithms are expected to show different accuracy for different processing units. The one with the highest accuracy would prove that the efficiency of the customer segmentation problem has been increased by employing a different method than the traditional one.

4 Implementation Detail

4.1 Dataset

The dataset employed for the training of the model is list of e-commerce transaction of 4000 customers spanning over one year's time period. The work that will be carried out is training the model using various algorithms in CPU and getting their accuracy while also implementing the said algorithms in GPU. This brings out a contrast between the two processing units as speedup of both are compared. Different forms of clustering and classifiers are employed for the process of customer segmentation.

4.2 Algorithms Used for Classification

The work first deals with classifying the customers in multi-core CPU using the following algorithms. The training model is fitted using various algorithms.

A. Support Vector Machine (SVM)

The Support Vector Machine (SVM) is a widely used classifier when talking about a recommender system. SVM uses the concepts of hyperplanes which divides sets of datapoint allowing it to be classified in different classes based on various attributes. The support vectors are essential the points which determine the margin of the hyperplane, the wider the hyperplane, more is the confidence with which the datapoints can be classified [23][25]. The work proposes the use of python in order to implement this algorithm for the given dataset.

B. Logistic Regression

The use of Logistic Regression is a highly mathematical one, which uses the logistic function in order to fit the model. The function takes various inputs in terms of weights and coefficients. Many types of logistic regression concepts are present, namely, binary logistic regression- providing only 2 values, ordinal logistic regression- providing output for data with ordered values [28][29]. The multinomial logistic regression takes in various attributes and provides the required output; thus, the work proposes the use of the same.

C. K-Nearest Neighbor (KNN)

K-Nearest neighbor is a simple concept which employs the concept of observing the neighboring data in order to classify any particular set of inputs. If K=1, the only a single neighbor

is checked and based on the class category of that neighbor, the input is classified [17][18]. In case of K=3, three of the neighbors would be cross-checked and corresponding class would be assigned. This is an efficient method if a proper value of K can be estimated.

D. Decision Tree

Decision Tree is one of the most widely used algorithms while being one of the simplest ones. It forms a tree by taking in various inputs, a tree that can be traced in order to make class decisions for a new input. It employs the top-to-bottom approach, and if a new input is taken in, then starting from the root it guides the user down to the leaf node where the final output of the class is located. More the attributes, the more complex will be the tree but it is still one of the simplest algorithms to implement [19].

E. Random Forest

Random forest classifier is an upgradation of decision tree. In simpler words, it is a combination of decision tress, thus, can be used for complex and intensive problems [20][21]. The way the class prediction is computed is by observing the number of trees that have produced the same class. The class that has been calculated by most decision tree is the one that is given to the new observation.

F. AdaBoost Classifier

AbaBoost Classifier is more of a performance checker classifier [22][23]. It uses the concept of fitting multiple models so that each subsequent model can remove the error of the previous model. This allows for an efficient prediction once the computation is completed [24]. AdaBoost requires an algorithm to be implemented along with it, and the most common one happens to be the decision tree.

G. Gradient Boost Classifier

The Gradient Boosting Classifier is an improvement over the AdaBoost classifier as it is more flexible when it comes to the loss function provided to it. The loss function is the one that needs to be optimized by the boosting method [26][27]. While AdaBoost works with particular loss function, the Gradient Boosting Classifiers do not have such constraints. Once the training model is implemented, the testing of prediction takes place, which gives the user the various accuracy of the above algorithms.

4.3 Accuracy Metrics

Precision can be defined as the ratio of the actual correct output to the total output that has been calculated. In terms of a recommender system, it can be measured as the ratio of the number of customers that are correctly segmented – true positive – to a particular attribute to the number of total customers that have been segmented – true positive plus false positive as shown in (1). A higher precision signifies that the segmentation that has taken place is very accurate. The value of the precision lies between 0 and 1, where 1 signifies that all the segmented customers are correctly segmented, while 0 denotes that none of the customers are correctly segmented.

$$precision = \frac{true\ positive}{true\ positive + false\ positive} \quad (1)$$

Recall or Sensitivity can be defined as the ratio of the number of positive results received to the total number of positive results there in existence as shown in (2). In terms of customer segmentation, it can be defined as the ratio of the total number of customers that have been correctly segmented based on an attribute to the total number of customers that should have bee segmented based on that attribute. A recall of 1 signifies that all the positive result has been correctly evaluated as positive, while a value of 0 signifies that none of the positive values in existence have been termed as positive.

$$recall\ (sensitivity) = \frac{true\ positive}{true\ positive + false\ negative} \quad (2)$$

F1 score can be defined as the accuracy metric for uneven distribution of the positive and negative values as shown in (3). F1 score takes both precision and recall into consideration and gives a metric value that is sometimes more useful than just the precision or recall of the model.

$$F1\ Score = \frac{2*(Recall*Precision)}{(Recall+Precision)} \quad (3)$$

Specificity can be defined as the ratio of the actual negative result that was received to the total number of negative results that exist in the received output as shown in (4). In terms of recommender system, it can be defined as the ratio of the total number of customers that did not get segmented to a particular group to the total number of customers that should not have get segmented to the same group. A Specificity of 1 signifies that all the segmentation that should not have taken place did not take place, whereas, a Specificity of 0 signifies that all the segmentation that should not have taken place did take place.

$$specificity = \frac{true\ negative}{true\ negative + false\ positive} \quad (4)$$

4.4 CPU and GPU

The Central Processing Unit (CPU) has been and will remain the most widely used processing system in the digital world. It is still the most optimized system for small tasks and updates to the number of cores and the way they are arranged have brought about massive applications for the same. The main problem that surrounds CPU is the thermal aspect, as even though the transistors have reduced in size, maintaining a billion transistor without it generating massive amount of heat is difficult. Multicore CPUs have come into play which allows for a reduced latency. CPU is efficient when fast processing of data is required, that is, it has a higher clock speed.

The Graphic Processing Unit (GPU) is the system with high processing power. It cannot be thought of as a separate entity that replaces CPU altogether but it is something that complements the same. The initial purpose of GPUs was the creation of graphic cards for video games but then it was evolved to handle excessive and massive data. Unlike CPU, which does a particular task in an efficient amount of time, GPUs can perform multiple tasks with numerous amounts of data in a more efficient manner. It follows the Single Instruction Multiple Data architecture, where it is equipped to handle huge amount of data concurrently, thus, as a whole

becomes more efficient than GPU, that is, it has a lower clock speed with huge number of cores.

Talking about parallelism in both, we see that CPU can be parallelized using two, three or four cores. GPU, on the other, hand has numerous cores say fourteen processors each having 32 cores. This makes concurrent tasks much more efficient and easier. CPU is known to be task oriented, as it has a high clock speed. It can complete a single task faster than a GPU. GPU is considered to be data oriented as it can take huge excessive data and work it concurrently giving efficient output. In case of CPU, each thread is individually programmed as the tasks run on different instructions. GPU programming can be done for batches of threads as same instructions are carried out in different data.

4.5 Parallelization Within Cores

A. Possible Advantages

The advantages of parallel programming lie in its feature of allowing concurrent processing of various instructions and taking multiple data at a single time. It executes the code efficiently does allows for subsequent processing of huge data and large problems in record time. While sequential programming executes one instruction at a time, the speedup of the entire system becomes very huge. In case of customer segmentation, many customers might need to be segmented into various categories simultaneously.

A particular customer might be part of an attribute that is defined by the likeness of "horror" shows while also being a part of the attributes that is defined by the likeness of "comedy" shows. If all these segments are created one at a time, the efficiency of the recommender system would reduce drastically. Parallel computation allows for all these segments to be created in a much more efficient manner, thus, making the recommender system useful for the users as well as the marketeers. Also, note that it is much more expensive to accommodate billions of transistors for sequential programming which also contributes to excessive thermal issues. Parallel programming makes this problem obsolete by the development of multi-core technologies.

B. Possible Drawbacks

No system is truly amiss from its disadvantage. A much newer system, as compared to sequential programming, parallel programming does pose certain difficulties to its implementation. Gaining insights on how to go about it becomes difficult as it has not been introduced every single field of computation where sequential programming dominates. Parallel programming in essence is a add on to sequential programming does it has certain drawbacks that sequential programming carries, but mostly those get taken care of as it further progresses. The main problem might that of deadlocks and the control of non-deterministic algorithms in these parallel systems.

Being based on so many different models, it may happen that the implementation becomes a bit complicated as compared to sequential programming. The tools that are used to implement sequential programming are widely available whereas the tools to facilitate parallel programming may be scarce. Despite of these drawbacks, parallel programming is still considered as one of the most efficient systems as it progresses and improves with more implementations being carried out on its base.

C. Amdahl's Speedup Law

Speedup can essentially be defined as the ratio of the time taken for a job to executed by a single sequential processor to the time taken for a job to executed by a number of, say N, parallel processors. When calculating the different speedups of different algorithms that are parallelized, we see by how many times does the execution speed increases. Amdahl came up with an equation known as the Amdahl's Speedup Law. The Amdahl's law is shown in (5), where serial execution time is equal to the time taken by one core of CPU, and parallel execution time is the time taken by multiple CPU and multiple GPU cores for execution of the same block of codes.

$$speedup = \frac{serial\ exeuction\ time}{parallel\ execution\ time} \quad (5)$$

Let's say that 90% of the implementation can be parallelized, and 5 processors are used, then approximately the implementation will be 3.6 times faster in parallelized form than it was in sequential. Similarly, if a piece of code when executed sequentially if takes 10ms and the same when executed concurrently over multiple cores, takes 5ms, then the speedup that is achieved, from (5), is 2 times. The algorithm which gives the highest speedup is the most efficient one.

Amdahl's law basically gives us the maximum theoretical speedup that can be achieved. In practicality, the speedup is less than the one equated by the given equation. The paper uses Amdahl's Speedup Law to calculate the speedup of various algorithms as they are parallelized in CPU and GPU to get the one which is most efficient.

4.6 Python Parallelization Library, Ray

Ray is a distributed framework and a Python library which facilitates the use of parallel programming by implementing its features. It is an efficient share-memory library which allows for the usage of excessive data. Ray can be used to parallelize the CPU while also allows for the usage of GPU. Ray can parallelize the CPU and GPU the number of both are specified by the users. Ray also has the ability to parallelize a particular function of the python code by itself, if the proper commands are provided.

The function that needs to be parallelized can be initialized by @ray.remote() which takes the number of CPU and GPU as input. Ray platform is a very efficient method to implement parallelization of both CPU and GPU in python framework. The framework parallelizes the tasks automatically over the many cores of GPU by CUDA programs for python, which is implicitly called based on the functions. The part of the code, which needs to be executed concurrently, are put inside a function that is preceded with a ray annotation to declare that the part must be parallelized.

5 Results and Analysis

5.1 System and Dataset Overview

Kaggle is an online platform that allows the users to find multiple datasets that can be used for the testing of their algorithms. Kaggle allows the coding of Python and R scripts. It provides with Kaggle Kernels which makes sharing of code easier. The dataset used is an e-commerce database. It includes purchases that have been made by 4000 customers over a span of one year. The columns of the dataset include InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID and Country.

5.2 Accuracy Metric Scores

Table 1 displays the accuracy metric scores of all the algorithms that have been implemented on the given dataset. The table can be used to get the basic idea as to how effective a particular algorithm for the given dataset. It shows the users which algorithm can be used to fit the model based on what is the requirement of the entire system.

TABLE 1
ACCURACY METRIC SCORES OF VARIOUS CLUSTERING ALGORITHMS FOR TRAINING THE DATASET

Algorithm	Precision	Recall or Sensitivity	F1 Score	Specificity
SVC	0.659728	0.675763	0.667649	0.762755
Logistic regression	0.743397	0.679210	0.709855	0.747674
k-NN	0.718528	0.722940	0.720727	0.677716
Decision Tree	0.622248	0.661917	0.699201	0.651446
Random Forest	0.740935	0.685363	0.652282	0.773665
AdaBoost	0.666095	0.659686	0.662875	0.697140
Gradient Boosting	0.620513	0.732850	0.672020	0.685315

As shown by the table, the highest Precision value is obtained by the Logistic Regression, a value of 0.7433. On the other hand, the lowest precision obtained by the Gradient Boosting with a value of 0.6205. All the other algorithms have a precision value within a range of 0.62 to 0.743. Now, if the system's requirement was how precise the trained dataset be then Logistic Regression would be the best fit as it is the most precise. Out of all the positive results it obtained, most were correctly identified as positive, thus, being the most precise.

Similarly, if the Recall values or the sensitivity values are noted, the highest value goes up to 0.7328 as obtained by Gradient Boosting, and the lowest value is that of 0.659 as obtained by AdaBoost Classifier. A system that focuses more on sensitivity should be using Gradient Boosting for the e-commerce dataset provided. Out of all the available positive data, most positive data are obtained. This defines the sensitivity of the code; hence, the Recall value and the Sensitivity values are the same.

F1 score helps in bringing out a middle ground in terms of the use of Recall and Precision. A particular system might be concerned with both Precision and Recall. This is where F1 score comes in. As is evident by the table, the highest F1 score is obtained by K-Nearest Neighbor, with a value of 0.720, while the lowest one is obtained by Random Forest with a value of 0.6522. F1 score is most often considered as the deciding factor as it gives a most accurate value.

Specificity helps in defining the true negative values out of all the values that have been considered to be negative. This also helps in determine how specific a particular algorithm is.

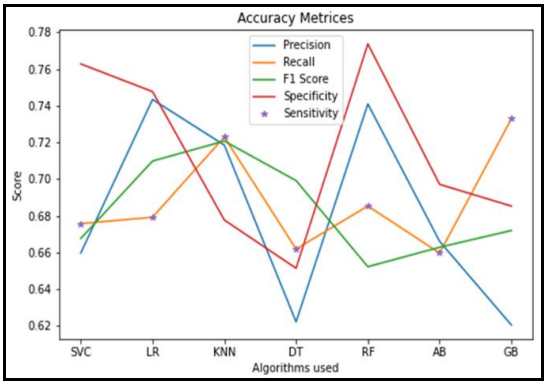

Fig. 3. Visualization of Accuracy Metrics for Various Algorithms.

Fig. 3 shows the visualization of Table 1. It can be seen that even though the range of score values of each accuracy metric is small, it makes a huge difference as 0.1 change accounts for 10 % change in the accuracy. In case of Precision it is seen that it has a zig-zag pattern with reaching a peak and then dropping with subsequent change in algorithm. It increases drastically from the algorithm SVC to Logistic Regression, then moves to decrease for K-Nearest neighbor dropping to its current lowest for Decision Tree. There is again a sudden peak at Random Forest which then declines for AdaBoost coming to its lowest of the lot for Gradient Boosting.

For Recall and Sensitivity, the zig-zag pattern can be observed again but the range is small as compared to that of the precision. It even has approximately stable values for SVC and Logistic Regression. It peaks at K-Nearest Neighbor and Random Forest but reaches the top for Gradient Boosting. Decision Tree and AdaBoost both see a drop in the recall values. F1 score, the one which is utilized the most, has an initial bell-shaped graph, which drops and peaks slightly at the end. F1 score sees its highest peak for K-Nearest Neighbor and lowest for Random Forest. Decision Tree and Logistic Regression show a similar trend with the F1 score.

Specificity has an initial decline for the first four algorithms, that is, SVC, Logistic Regression, K-Nearest Neighbor and Decision Tree. It surges up to record high for Random Forest and then again drops down for AdaBoost and Gradient Boosting, while Decision Tree still seeing the lowest value. For all the accuracy metrics, Decision Tree and AdaBoost is seen to perform the worst. If F1 score is considered to be a fair evaluator then K-Means Clustering is the algorithm that should be used for the e-commerce dataset.

5.3 Parallelization Observation

Parallelization is basically a process that allows different instructions to work simultaneously while handling excessive data. It allows for concurrent execution of different instructions; thus, it decrease the execution time considerably. It is of utmost importance as the current world demands faster

and more efficient execution systems and parallelization is the answer to that. Table 2 shows the execution time of each system to implement the given algorithm to the e-commerce dataset. The process of parallelization has been implemented; hence, multi-core CPUs have been used, in order to show the

TABLE 2
EXECUTION TIME TAKEN OVER MULTI CORES OF CPUS AND MANY CORES OF GPU FOR VARIOUS CLUSTERING ALGORITHMS

Algorithm	1-Core CPU Time (s)	2-Cores CPU Time (s)	3-Cores CPU Time (s)	4-Cores CPU Time (s)	GPU Time (s)
SVC	357.36	306.47	254.83	223.88	68.95
Logistic regression	343.98	318.32	261.70	214.80	93.01
k-NN	375.33	301.28	243.81	200.02	62.23
Decision Tree	347.54	302.78	270.40	228.96	119.9
Random Forest	368.22	317.85	268.62	256.70	116.3
AdaBoost	366.03	300.70	254.17	257.25	114.1
Gradient Boosting	344.06	310.31	272.30	258.96	60.70

comparison. The GPU system has also been used to parallelize in order to show the drastic difference between CPU and GPU. From Table 1, it can also be observed that the execution time when 1-Core CPU is used is expected to be the most as it is basically a sequential execution of each algorithm. The most time taken is by K-Nearest Neighbor, that of, 375.33 s, and the least time is taken by Logistic Regression, that of 343.98 s. KNN algorithm seems to be most unsuitable choice as an algorithm for the required system. As parallelization is thrown into the mix, a slight difference in the execution time is seen. The execution time starts decreasing with slight differences as the number of cores of CPU increases. For 2-cores CPU, the least time is taken by AdaBoost, that of 300.70 s, while the most time is taken by Logistic Regression, 318.32 s. It can be observed that even the algorithm taking the most time is the less than the algorithm taking the least time in a sequential 1-core CPU. For 3-Core CPU, we see further progress as the even the highest execution time becomes 272.30 s, that of Gradient Boosting which is considerably less than any algorithm previously visited for lesser number of cores. The least value goes down to 243.81 s for K-Nearest Neighbor. The 4-cores CPU show the best result out of the four with the most execution time of 258.96 s for Gradient Boosting while the least that of 200.02 for K-Nearest Neighbor. It can be seen that when the cores of CPU are compared, on an average, a decrease of 1.5 times of the execution time is seen from the sequential 1-core CPU execution.

Using GPU for the same parallelization yields tremendously positive results. The highest execution time goes as low as 119.9 s for Decision Tree. A value so low is less than two times that of the sequential programming. When its lowest value is observed, it shows an astonishing result of 62.23 s for K-Nearest Neighbor, that has the highest execution time in case of sequential programming. Thus, the algorithm which was the most unsuitable for sequential programming became the most suitable for parallel programming in GPU.

Fig. 4 shows the visualization of the execution time that each multi-core CPU takes to implement a given algorithm. If seen in a vertical manner, it can be observed that for each algorithm, as the number of cores of CPU increases the execution time decreases.

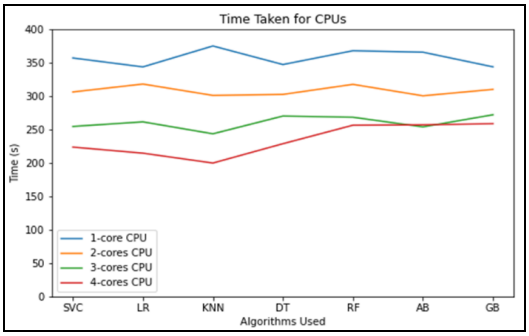

Fig. 4. Visualization of Time Taken for Execution Over Single Core and Multi Cores of CPU.

From Fig. 4, it can also be observed that there is just one discrepancy in case of AdaBoost Classifier which has a slightly greater execution time for 4-cores CPU than 3-cores CPU, which is caused by parallel overheads.

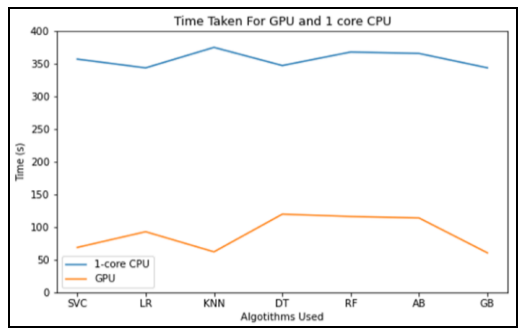

Fig. 5. Visualization of Time Taken for Execution Over Single Core of CPU and Many Cores of GPU.

Fig. 5 shows the drastic difference between a sequential 1-core CPU programming to the parallel GPU programming. The execution time for each algorithm mentioned is drastically low as compared to the execution time as obtained by sequential 1-core CPU.

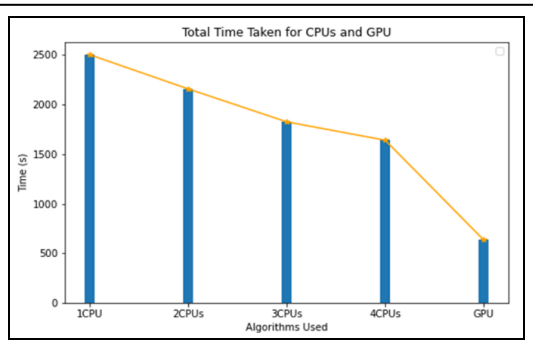

Fig. 6. Visualization of the Total Execution Time Over Single and Multi Cores of CPU and Many Cores of GPU.

Fig. 6 shows the total execution time that was taken by CPU and GPU to execute all the algorithms on the dataset in

sequential and parallel mode. The graph shows a declining slope indicating that 1-core CPU takes the most time to implement any instruction, while GPU, which is multi-core, takes the least time to execute an instruction when it is parallelized.

The graphs shown in Fig. 4, Fig. 5 and Fig.6 proves that it is better to parallelize using 4-cores CPU as the execution time decreases as compared to the sequential 1-core CPU. It can also be noticed that an approximate 6 times increase is seen in the execution time of 1-core CPU as compared to the multi-core GPU. This proves that it is beneficial to use GPU for parallelization as it produces tremendously desired effects.

TABLE 3
SPEEDUP OBTAINED BY VARIOUS PARALLEL SYSTEMS

Algorithm	2-Cores CPU	3-Cores CPU	4-Cores CPU	GPU
SVC	1.166044	1.402366	1.596248	5.182684
Logistic regression	1.080625	1.314421	1.601439	3.698390
k-Nearest Neighbors	1.245783	1.539464	1.876476	6.031394
Decision Tree	1.147812	1.285283	1.517889	2.898296
Random Forest	1.158471	1.370780	1.434426	3.165313
AdaBoost	1.217249	1.440112	1.422845	3.208136
Gradient Boosting	1.108776	1.263547	1.328629	5.668426
Average Speedup	1.160680	1.373710	1.539707	4.264663

5.4 Speedup Obtained

Amdahl's Speedup Law is used to predict the maximum speedup that can be achieved by a particular system. The speedup defines how fast or efficient is a system as compared to other.

Table 3 shows the speedup achieved by different systems, that is, the multi-core CPUs and GPU as compared to the 1-core CPU. It is basically the ratio of the time taken for an instruction to be executed by a single processor system to the one with multi-processor. It can be seen from the table that speedups obtained by the multi-core CPUs are less than 2 times that of 1-core CPU, with the maximum being 1.87 times for K-Nearest Neighbor in 4-cores CPU. When the speedups by GPU is observed we see that the lowest increase is still that of 2.89 times for Decision Tree while the highest goes up to 6.03 times for K-Nearest Neighbor.

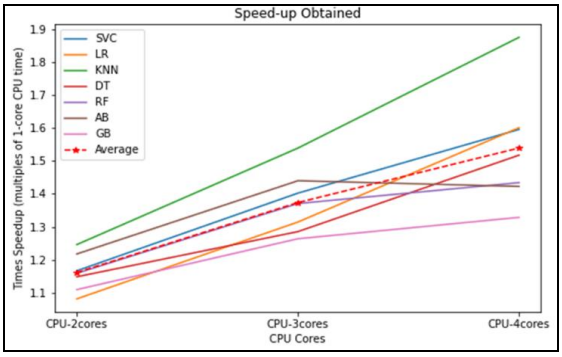

Fig. 7. Visualization of the Speed-up obtained for CPU Cores.

Fig. 7 shows that the speedup increases for all the algorithms with increased number of CPU cores. For k-Nearest Neighbors algorithm, the speedup is the most, for Gradient Boost algorithm, the speedup increases but not as significantly as others. However, for AdaBoost algorithm, the speedup decreases when four cores of CPU are used instead of three cores, which shows the parallel overheads that must have been created due to inter-core communications. The average speedup, on the other hand is decently increasing, in almost a straight line.

Number of cores in GPU in comparison with CPUs are huge, thus the speedup is also very high in comparison with that of a single core CPU, and the same can be observed in Table 3. On average, GPU has a speedup of 4.264, that is, it is 4.26 times faster than a single-core CPU. This is caused by the huge number of cores present in GPU, which is much more than four. Thus, the table shows that for the given dataset, it is most optimized to use K-Nearest Neighbor algorithm which is implemented by parallelizing GPU.

Finally, from the observation of time-taken and accuracies, it can be inferred that parallelization may help in obtaining speedup without affecting the overall accuracy of the systems, given the fact that the parallelization has been carried out in a restricted fashion, which will result in balanced grained tasks with minimum inter-process and inter-core communications and minimum parallel overheads. Increase in number of processors and cores can catalyze the performance and speed only when the inter-process communications as well as the task module dependencies are less, otherwise, increased number of processors can also act as inhibitors by adversely resulting in more time if the processors might have to wait for data which is blocked by some other processor, or might have to wait for a task to get executed.

6 CONCLUSION

Customer segmentation is the crux of any recommender system. Recommender Systems, on the other hand, have become the crux of online shopping and streaming platforms. Every online streaming platform bases its profits on the success of its recommender system, while a recommender system is only as good as its customer segmentation algorithms. In a digitized world like such, time has become the utmost treasure and everyone wants to interact with platforms and devices that do not take up lot of their time. If the recommendation system fails to provide the service in the optimal time then it loses its usefulness as a system.

With the advent of parallelization, execution time of any and every system has been drastically reduced. The system that has been created allows for customer segmentation to take place in a much more efficient manner. The juxtaposition of the multi-core CPUs and the GPU have clearly shown that parallelization of GPU allows for a much better and an efficient system. The average speedup of GPU with a value of 4.24 as compared to 1-core CPU shows the positive applications of parallel computing along with the use of GPU to implement the same.

The work can be enhanced in terms of future prospects by implementing and improving the same on datasets that have a

huge data set of online purchases or online streaming history, allowing for the system to learn and became a better version of itself.

REFERENCES

[1] KABASAKAL, İnanç. "Customer Segmentation Based On Recency Frequency Monetary Model: A Case Study in E-Retailing." *International Journal of InformaticsTechnologies* 13.1 (2020).

[2] Alkhayrat, Maha, Mohamad Aljnidi, and Kadan Aljoumaa. "A comparative dimensionality reduction study in telecom customer segmentation using deep learning and PCA." *Journal of Big Data* 7.1 (2020): 9.

[3] Cahyana, Bambang Eka, et al. "Hybrid cluster analysis of customer segmentation of sea transportation users." *Journal of Economics, Finance and Administrative Science* (2020).

[4] Carnein, Matthias, and Heike Trautmann. "Customer segmentation based on transactional data using stream clustering." *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, Cham, 2019.

[5] Hung, Phan Duy, Nguyen Thi Thuy Lien, and Nguyen Duc Ngoc. "Customer segmentation using hierarchical agglomerative clustering." *Proceedings of the 2019 2nd International Conference on Information Science and Systems*. 2019.

[6] Datta, Debajit, et al. "Comparison of Performance of Parallel Computation of CPU Cores on CNN model." *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*. IEEE, 2020.

[7] Karahoda, Sertaç, et al. "Multicore and manycore parallelization of cheap synchronizing sequence heuristics." *Journal of Parallel and Distributed Computing* 140 (2020): 13-24.

[8] Gerzhoy, Daniel, et al. "Nested mimd-simd parallelization for heterogeneous microprocessors." *ACM Transactions on Architecture and Code Optimization (TACO)* 16.4 (2019): 1-27.

[9] Kim, Sang Hee, et al. "Computing Performance Comparison of CPU and GPU Parallelization for Virtual Heart Simulation." *Journal of Biomedical Engineering Research* 41.3 (2020): 128-137.

[10] Essaid, Mokhtar, et al. "GPU parallelization strategies for metaheuristics: a survey." *International Journal of Parallel, Emergent and Distributed Systems* 34.5 (2019): 497-522.

[11] Rosenberg, Duane, et al. "GPU Parallelization of a Hybrid Pseudospectral Geophysical Turbulence Framework Using CUDA." *Atmosphere* 11.2 (2020): 178.

[12] Rodriguez, Mayra Z., et al. "Clustering algorithms: A comparative approach." *PloS one* 14.1 (2019): e0210236.

[13] Bai, BG Mamatha, B. M. Nalini, and Jharna Majumdar. "Analysis and detection of diabetes using data mining techniques—a big data application in health care." *Emerging Research in Computing, Information, Communication and Applications*. Springer, Singapore, 2019. 443-455.

[14] Zhang, Shuai, et al. "Deep learning based recommender system: A survey and new perspectives." *ACM Computing Surveys (CSUR)* 52.1 (2019): 1-38.

[15] Osadchiy, Timur, et al. "Recommender system based on pairwise association rules." *Expert Systems with Applications* 115 (2019): 535-542.

[16] Natarajan, Senthilselvan, et al. "Resolving data sparsity and cold start problem in collaborative filtering recommender system using linked open data." *Expert Systems with Applications* 149 (2020): 113248.

[17] Chen, Yewang, et al. "Fast density peak clustering for large scale data based on kNN." *Knowledge-Based Systems* 187 (2020): 104824.

[18] Ma, Chencheng, Xuehui Du, and Lifeng Cao. "Improved KNN Algorithm for Fine-Grained Classification of Encrypted Network Flow." *Electronics* 9.2 (2020): 324.

[19] Sarker, Iqbal H., et al. "Behavdt: a behavioral decision tree learning to build user-centric context-aware predictive model." *Mobile Networks and Applications* 25.3 (2020): 1151-1161.

[20] Katuwal, Rakesh, Ponnuthurai Nagaratnam Suganthan, and Le Zhang. "Heterogeneous oblique random forest." *Pattern Recognition* 99 (2020): 107078.

[21] Zhang, Pin, et al. "A novel hybrid surrogate intelligent model for creep index prediction based on particle swarm optimization and random forest." *Engineering Geology* 265 (2020): 105328.

[22] Xing, Hong-Jie, and Wei-Tao Liu. "Robust AdaBoost based ensemble of one-class support vector machines." *Information Fusion* 55 (2020): 45-58.

[23] Sun, Jie, et al. "Class-imbalanced dynamic financial distress prediction based on Adaboost-SVM ensemble combined with SMOTE and time weighting." *Information Fusion* 54 (2020): 128-144.

[24] Wu, Yanli, et al. "Application of alternating decision tree with AdaBoost and bagging ensembles for landslide susceptibility mapping." *Catena* 187 (2020): 104396.

[25] Hu, Rongyao, et al. "Robust SVM with adaptive graph learning." *World Wide Web* 23.3 (2020): 1945-1968.

[26] Yu, Bin, et al. "SubMito-XGBoost: predicting protein submitochondrial localization by fusing multiple feature information and eXtreme gradient boosting." *Bioinformatics* 36.4 (2020): 1074-1081.

[27] Zhang, Yanju, et al. "PeNGaRoo, a combined gradient boosting and ensemble learning framework for predicting non-classical secreted proteins." *Bioinformatics* 36.3 (2020): 704-712.

[28] Feng, Yunlong, Jun Fan, and Johan AK Suykens. "A Statistical Learning Approach to Modal Regression." *Journal of Machine Learning Research* 21.2 (2020): 1-35.

[29] Guo, Yanrong, Zhengwang Wu, and Dinggang Shen. "Learning longitudinal classification-regression model for infant hippocampus segmentation." *Neurocomputing* 391 (2020): 191-198.

[30] Jagani, Khyati, Falguni Vasavada Oza, and Himani Chauhan. "Customer Segmentation and Factors Affecting Willingness to Order Private Label Brands: An E-Grocery Shopper's Perspective." *Improving Marketing Strategies for Private Label Products*. IGI Global, 2020. 227-253.