Accelerating Airline Delay Prediction-Based *P-CUDA* Computing Environment

Dharavath Ramesh, Neeraj Patidar, Teja Vunnam and Gaurav Kumar

Abstract Machine learning techniques have enabled machines to achieve human-like thinking and learning abilities. The sudden surge in the rate of data production has enabled enormous research opportunities in the field of machine learning to introduce new and improved techniques that deal with the challenging tasks of higher level. However, this rise in size of data quality has introduced a new challenge in this field, regarding the processing of such huge chunks of the dataset in limited available time. To deal such problems, in this paper, we present a parallel method of solving and interpreting the ML problems to achieve the required efficiency in the available time period. To solve this problem, we use CUDA, a GPU-based approach, to modify and accelerate the training and testing phases of machine learning problems. We also emphasize to demonstrate the efficiency achieved via predicting airline delay through both the sequential as well as CUDA-based parallel approach. Experimental results show that the proposed parallel CUDA approach outperforms in terms of its execution time.

Keywords Machine learning (ML) · Naïve Bayes · GPU · CUDA Tree reduction

D. Ramesh (✉) · N. Patidar · T. Vunnam · G. Kumar
Department of Computer Science and Engineering, Indian Institute of Technology (ISM), Dhanbad 826004, Jharkhand, India
e-mail: ramesh.d.in@ieee.org

N. Patidar
e-mail: neerajism@cse.ism.ac.in

T. Vunnam
e-mail: vunnamteja@gmail.com

G. Kumar
e-mail: gaurav315kumar@gmail.com

D. Reddy Edla et al. (eds.), *Advances in Machine Learning and Data Science*,
Advances in Intelligent Systems and Computing 705,
https://doi.org/10.1007/978-981-10-8569-7_2

1 Introduction

The introduction of new technologies has made the computer science at its best as an emerging field. With the availability of new software paradigms, dependency on computers and machines to perform extreme and computationally exhaustive tasks is growing day by day. For example, the traditional way of file storage on local file systems is replaced by distributing and even more secure cloud storage systems. Among all, the Internet has turned out to be the most resourceful technology ever created for instant sharing of knowledge and resources with the rest of the world with ease. Its popularity around the world can be assessed by the fact that the total number of Internet users has grown by around 82% or almost 1.7 billion in the last 5 years and this number is forecasted to increase up to 4 billion around the year 2020. The main reasons behind this increasing popularity of the Internet are its speed, economic nature and ease of accessibility to the users. The aid of Internet has made the activities easier to a tremendous extent by reducing the communication delay between users in the different parts of the world.

The dependency and conscience to improve our methods by means of research and analysis have led to an immense hike in the rate of data production. As a result of its increased popularity, around 90% of the world's data has come into existence in the past 2 years and Internet has turned out to be the largest contributor among its sources. With the availability of sufficient data for research and analytics, machine learning and related techniques have picked up popularity [1] to achieve some computationally impossible solutions by using well-defined mathematical models. Machine Learning [2] is a term used to define a technique to find a solution or more precisely improve the existing solution gradually by following the process of learning through previous observation or experience without any human intervention. Machine learning is a popular community to solve problems which require human-like instinct and decision-making. With the ease and availability of Big Data, machine learning is used to solve some complex and interesting problems [1] which help in achieving those tasks that normally require some special human assistance, intelligence or decision-making skills to execute successfully. For example, making classification, predictions in advanced robotics, and driver-less cars include such tasks.

1.1 Limitations of Classical ML Methodology

The basic approach to solve the machine learning problem involves two subtasks: (i) training and (ii) testing. Apart from the techniques [1] used, both these subtasks are common in the process of solving any machine learning task. Both these tasks are extremely crucial in solving a machine learning problem. The quality of the training and testing methods eventually decides the overall quality of the model or solution. Initially, training is performed to train or develop a mathematical model/classifier/machine with the help of existing chunks of data in a huge quantity. The

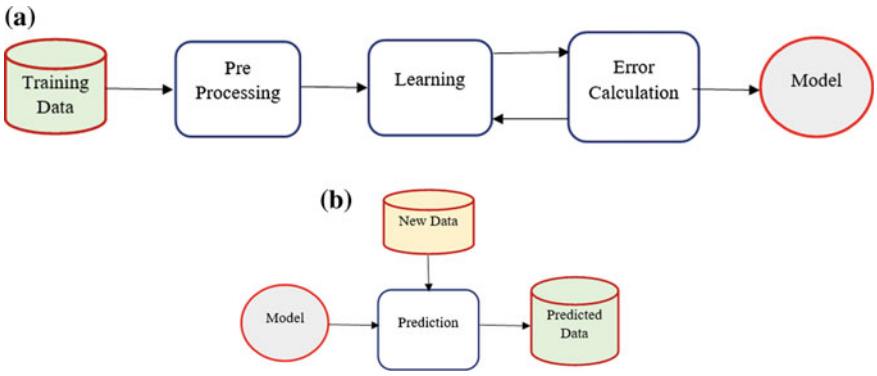

Fig. 1 Machine learning as a process, **a** learning phase, **b** testing phase

vast amounts of data help in improving the accuracy and performance of the model, but processing these chunks of data is another crucial problem. Hence, the overall performance of the ML algorithm is limited by the processing methodologies used to process the data for training the model. The same can be improved to process those huge datasets, so that appreciable results can be achieved within the conserved time limits. A learning phase and training phase of ML process are depicted in Fig. 1a, b.

The second task is testing and analysing the accuracy of the model by using the related test data. This step is crucial as it measures the accuracy and the acceptability of the model. But, the solution must work for regular as well as new problems [2], i.e. to make the solution robust against different possible situations. A suitable ML model can be used to test the model against a large dataset which may or may not belong to dataset. Again, the vastness of the dataset will extend the evaluation period of our model which is completely undesirable. To deal with these kind of problems, several efforts have been made in this field to improve its performance. To over-relate and achieve this problematic instance, in this paper, we discuss a strategy using the GPU-based parallel processing platform named as CUDA, to perform the ML tasks in allowable time limits [3, 4]. This approach works in an efficient way and can make the training and testing strategies workable within the time period scan.

2 Resolving with CUDA Platform

2.1 CUDA

In 2007, NVIDIA, a GPU-designing firm, released the initial version of CUDA—a parallel computing platform which uses the graphical processing units (GPUs) for massive parallelization of computation tasks [4]. It follows general purpose

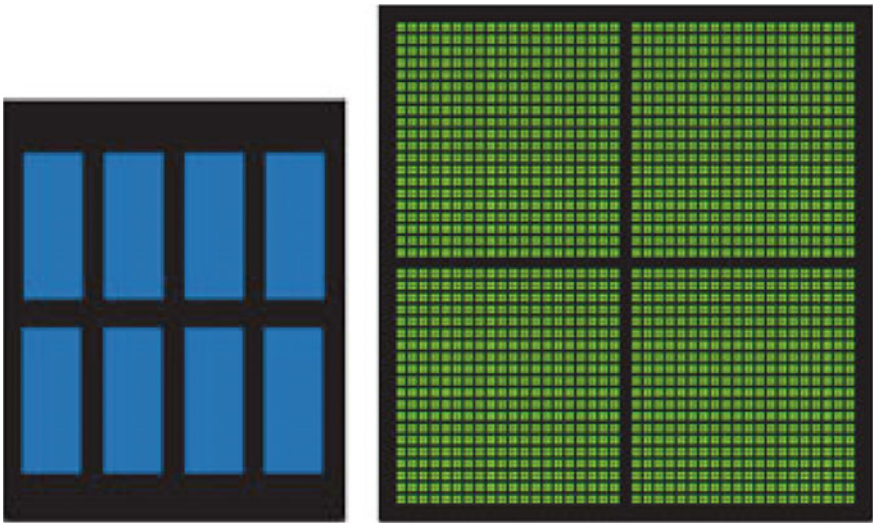

Fig. 2 CPU versus GPU

computing, on graphics processing unit (GPGPU) approaches to perform the general tasks which are computationally intensive and require a lot of time to be executed if performed on the CPU [5]. It allows users to write massively parallel programs using languages like C, C++ and Fortran [6]. The CPUs are optimized for sequential processing and contain only limited numbers of powerful cores, whereas on the other hand, GPUs are optimized for performing parallel tasks and contain thousands of smaller and less powerful cores [7]. The graphical representation of CPU and GPU is depicted in Fig. 2.

2.2 Impact on Solving ML Problems

Solution to the ML problems using traditional systems may turn out to be an expensive technique with the increase in the size of the dataset. Especially, the time required to process such a large set of data increases drastically. Using parallel GPU, computation can turn out to be a perfect solution for solving such large-scale machine learning problems without any extra increase in the machine cost by harnessing the graphical computation power of the system [3, 8].

3 Problem Formulation: Predicting Flight Delay Occurrences

Every year around one-fifth of the airline flights suffer a cancellation or delay which results in critical loss of time and resources to both airlines and passengers. Hence, creating a model which can predict whether a given flight with certain parameters like source, destination, duration will suffer a delay or not can offer huge help to the passengers and airline managers in choosing and managing their flights. Here, we use a dataset from American Statistical Association (ASA) 2009 data expo [9], which includes dataset for all commercial flights within USA from year 1987 to 2008. This dataset includes nearly 120 million flight records. To predict the future occurrences, we consider Naïve Bayes Classifier (NBC). The sequential standard of NBC is described in Sect. 3.1.

3.1 Solution: Sequential Naïve Bayes Classifier

3.1.1 Naïve Bayes Classifier

Naïve Bayes Classifier (NBC) belongs to a class of conditional probabilistic classifiers which assume a major assumption of independence between all features of its dataset to calculate most probable outcome depending on the input given by the user [10]. It uses Bayesian interpretation to measure a degree of belief to identify the class which includes the input features [11]. Given an input in the form of a vector;

$$X = (x_1, x_2, x_3, \ \ldots, x_n) \tag{1}$$

The NBC will provide the probability P_k for each class, whether the input vector X belongs to the given class C_k, i.e.

$$P(C_k \mid (x_1, x_2, x_3, \ \ldots, x_n)) \tag{2}$$

Using Bayes theorem, the given probability can be decomposed as;

$$P(B \mid A) = \frac{P(B)P(A \mid B)}{P(A)} \tag{3}$$

The above formulae can be extended to calculate the joint probability considering several parameters for each sample;

$$P(B_i \mid A) = \frac{P(B_i)P(A \mid B_i)}{\sum_{i=1}^{n} P(B_i)P(A \mid B_i)} \tag{4}$$

However, the denominator can be ignored since it will remain same for all the classes [12]. Hence, the classification problem reduces to finding numerator for a given input vector *X*. Using the assumption of independence between the features of the vector *X,* i.e. the probability of occurrence of one feature, is independent of the probability of occurrence of any other feature or collection of features. At the same time, we can simplify the calculation of the numerator probability same as a joint probability.

3.1.2 Normal Distribution

For calculating the probability of occurrence of features which are continuing in nature rather the discrete, we use the normal (also known as Gaussian) distribution function [10] which can be given as follows.

$$P(x_k \mid C_i) = g(x_k, \mu_{Ci}, \sigma_{Ci}) = \frac{1}{\sqrt{2\pi}\sigma_{Ci}} e^{\frac{(x_k - \mu_{Ci})^2}{2\sigma_{Ci}^2}} \tag{5}$$

where $g(x_k, \mu_{Ci}, \sigma_{Ci})$ is the Gaussian (normal) density function for attribute A, while μ_{Ci} and σ_{Ci} are the mean and standard deviation, respectively, given the values for attribute A for C_i. Advantages of using NBC over other classifiers are: (i) simple and easy to test new data, (ii) minimum error rate as compared to other complex algorithms, (iii) high degree of accuracy with lower computational complexity and (iv) compete with more advanced algorithms like SVM, etc.

3.2 Methodology: Predicting Airline Delays

3.2.1 Training the Model

To train the model, we start by calculating mean and variance for each relevant parameter in the flight description which belong to either class $C_{Delayed}$ or $C_{Nodelay}$. The calculated values of mean and variance are shown in Table 1.

Using the table similar to above, we calculate the probabilities to find out the numerator for given values of features of flight whose class has to be identified as either delayed or on-time.

Table 1 Sample airline delay database

Sample	Dep. time	CRS dep. time	Arr. time	CRS arr. time	Act elapsed time	CRS elapsed time	Arr. delay	Dep. delay
1.	741	730	912	849	91	79	23	11
2.	729	730	903	849	94	79	14	−1
3.	741	730	918	849	97	79	29	11
4.	729	730	847	849	78	79	−2	−1
5.	749	730	922	849	93	79	33	19
6.	728	730	848	849	80	79	−1	−2
7.	728	730	852	849	84	79	3	−2
8.	731	730	902	849	91	79	13	1
9.	744	730	908	849	84	79	19	14
10.	729	730	851	849	82	79	2	−1

3.2.2 Testing

After performing preprocessing, we use mean and variance calculated in the previous step to find the posterior probabilities for the given test samples. After combining the posterior probabilities of all the features, we get the combined probabilities of that test sample belongs to the same class. For each sample, the class with the highest numerator value can be considered as the most optimum prediction for that sample. After this, we apply the strategy to find the probability for all the flight variables in the test dataset one by one, sequentially. The corresponding algorithm which includes these two steps is shown in Algorithm 1.

Algorithm 1: *NBC Classification algorithm*

Input: *Airline Dataset*
Output: *Set of probabilities for each class*

Step-1. Select the relevant features for each flight to be considered in the flight delay evaluation.
Step-2. Find **Mean ()** and **Variance ()** for the selected features over each class in the complete dataset.
Step-3. For each input vector, **X** calculate the individual probability of occurrence for each different class.
Step-4. Combine all the probabilities, to calculate the joint probability in the numerator.
Step-5. Select and assign each input vector class which has the highest numerator value.

4 Parallelizing the Approach

In this section, we test and improve the performance of the above algorithm using the parallel CUDA computing model [13, 14]. To accelerate the process to find the mean, we use a methodology known as *Tree-Based Reduction Sum*. This technique utilizes the parallel architecture of CUDA [6] to efficiently calculate the mean and variance for a given feature belonging to a particular class. The instance of *Tree-Based Reduction Sum* is depicted in Fig. 3.

In reduction sum, we particularly focus on the tree reduction technique to evaluate the sum of the given large set of values which can produce a speedup nearly equal to *log(n)*. As CUDA has no global synchronization facility to easily communicate with all the threads [13], this limitation can be avoided by using a recursive kernel invocation while adding a small hardware/software overhead in the individual kernel launch [15]. The state of recursive kernel invocation is shown in Fig. 4.

4.1 Parallelizing Testing Phase

To classify the test dataset, we select an optimum block size (i.e. the total number of threads per block) and total number of blocks each containing the threads equal to the block size [16]. After passing the set of mean and variance related to each

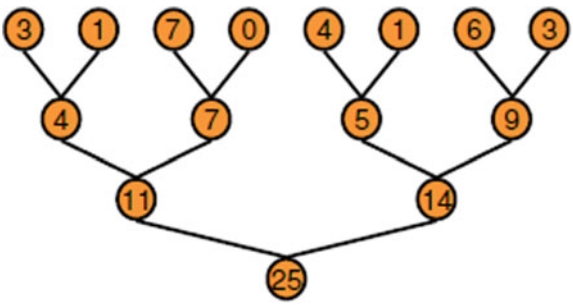

Fig. 3 Tree reduction sum approach

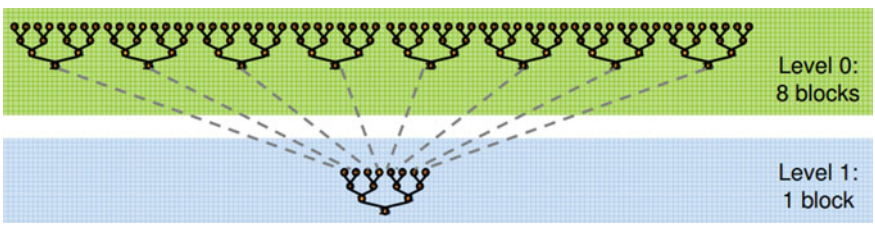

Fig. 4 Recursive kernel invocation

feature in the flight description, we calculate the individual probabilities of the test flight belonging to a particular class one by one for all possible classes [9]. Figure 5 shows the calculation through the NB model. Hence, each block processes the number of flights equal to the number of threads, i.e. one thread per test flight that evaluate the total results. As compared to the sequential processing of each test flight, the parallel CUDA algorithm only takes time equivalent to the number of threads per block and some communication time between the CPU and GPU to distribute [17] the process and recombine the solution. The modified parallel classification algorithm is shown in Algorithm 2.

Algorithm 2: *Parallel NBC Classification algorithm*

Input: *Airline Dataset*
Output: *Set of probabilities for each class*

Step-1. Select the relevant features for each flight to be considered in the flight delay evaluation.
Step-2. Find **Mean** () and **Variance** () using the reduction sum technique for the selected features over each class in the complete dataset.
Step-3. Divide the dataset into set of blocks which includes flight vectors equal to number of threads. For each input vector, **X** calculate the individual prob ability of occurrence for each different class.
Step-4. Combine all the probabilities, to calculate the joint probability in the numerator.
Step-5. Select and assign each input vector class which has the highest numerator value.

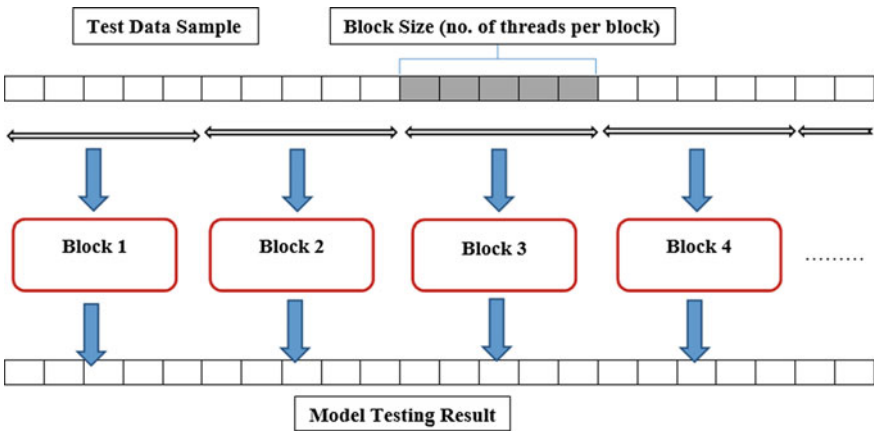

Fig. 5 Parallel test results calculation through NB model

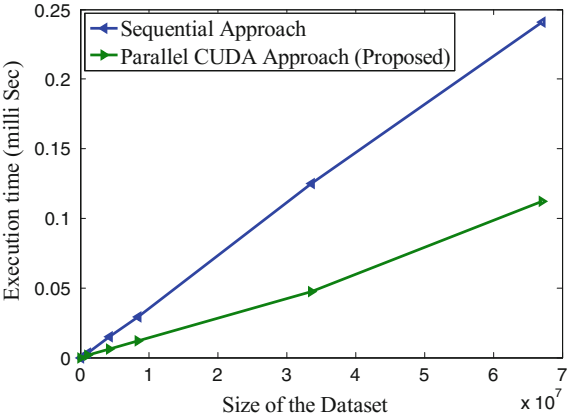

Fig. 6 Parallel CUDA implementation with the sequential method

4.2 Experimental Analysis

To perform the experiment-related approaches, we use the airline dataset of the American Statistical Association (ASA) [9]. The system on which we have conducted our tests has Windows 10 operating system running over Intel core i5-3337U processor 1.80 GHz with turbo boost up to 2.70 GHz and 6 GB of DDR3L internal memory. The GPU version is NVIDIA Geforce GT 740 M with dedicated 1 GB DDR3 memory. The coding and debugging of the CUDA project are done over Microsoft Visual Studio 2015 community edition with the latest CUDA toolkit 8.0 installed on the system. The compared results with the sequential algorithmic approach are depicted in Fig. 6.

From the results, we can conclude that the inclusion of the GPU support in training the model and testing the sample test data has resulted in a speedup which is nearly equal to 2.5 times the sequential methods. It also outperforms the related time execution sequences. However, this calculation may vary with different versions of NVIDIA GPU cards with different CUDA core configurations used for the purpose [18].

5 Conclusion and Future Work

This paper aims to describe how the enormous parallelizing power of CUDA platform can be utilized to accelerate the process of developing the ML model for prediction and the classification purpose. To make this scenario's performance better, we used the reduction sum technique to parallelize the task of calculation of the mean in the training phase and parallelized the testing process using the simple CUDA computation scheme. Both these techniques result in combined actual improvement over the sequential methodology to calculate the model and evaluate

the result. Similar improvements can be made in developing other ML models by utilizing the resources offered through CUDA computing platform to improve the time and resources used by overall prediction and classification process without using costly machines to process on it.

Acknowledgements This work is partially supported by Indian Institute of Technology (ISM), Government of India. The authors wish to express their gratitude and thanks to the Department of Computer Science and Engineering, Indian Institute of Technology (ISM), Dhanbad, India, for providing their support in arranging necessary computing facilities.

References

1. Liao, S.H., Chu, P.H., Hsiao, P.Y.: Data mining techniques and application—a decade review from 2000 to 2011. Expert Syst. Appl. **39**(12), 11303–11311 (2012)
2. Carbonell, J.G., Michalski, R.S., Mitchell, T.M.: An overview of machine learning. In: Machine Learning, pp. 3–23. Springer, Berlin, Heidelberg (1983)
3. Che, S., Boyer, M., Meng, J., Tarjan, D., Sheaffer, J.W., Skadron, K.: A performance study of general-purpose applications on graphics processors using CUDA. J. Parallel Distrib. Comput. **68**(10), 1370–1380 (2008)
4. Wu, R., Zhang, B., Hsu, M.: GPU-accelerated large scale analytics. IACM UCHPC (2009)
5. Ghorpade, J., Parande, J., Kulkarni, M., Bawaskar, A.: GPGPU processing in CUDA architecture (2012). arXiv:1202.4347
6. Farber, R.: CUDA Application Design and Development. Elsevier (2011)
7. Yang, C.T., Huang, C.L., Lin, C.F.: Hybrid CUDA, OpenMP, and MPI parallel programming on multicore GPU clusters. Comput. Phys. Commun. **182**(1), 266–269 (2011)
8. Harris, M.: Optimizing CUDA. In: SC07: High Performance Computing with CUDA (2007)
9. Data for experimentation, American Statistical Association, Data Expo (2009). http://stat-computing.org/dataexpo/2009/the-data.html
10. Murphy, K.P.: Naive Bayes Classifiers. University of British Columbia (2006)
11. Rish, I.: An empirical study of the naive Bayes classifier. In: IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence, vol. 3, No. 22, pp. 41–46. IBM, New York (Aug 2001)
12. Jia, P.T., He, H.C., Lin, W.: Decision by maximum of posterior probability average with weights: a method of multiple classifiers combination. In: Proceedings of 2005 International Conference on Machine Learning and Cybernetics, vol. 4, pp. 1949–1954. IEEE (Aug 2005)
13. Jian, L., Wang, C., Liu, Y., Liang, S., Yi, W., Shi, Y.: Parallel data mining techniques on graphics processing unit with compute unified device architecture (CUDA). J. Supercomput. **64**(3), 942–967 (2013)
14. Zhou, L., Wang, H., Wang, W.: Parallel implementation of classification algorithms based on cloud computing environment. TELKOMNIKA Indones. J. Electr. Eng. **10**(5), 1087–1092 (2012)
15. Fang, W., Lau, K.K., Lu, M., Xiao, X., Lam, C.K., Yang, P.Y., Yang, K. et al.: Parallel data mining on graphics processors. In: Technical Report HKUST-CS08-07. Hong Kong University Science and Technology, Hong Kong, China (2008)
16. Chengpeng, Y., Zhanchun, G., Yanjun, J.A.: GPU-based Native Bayesian algorithm for document classification. http://www.paper.edu.cn/lwzx/en_releasepaper/content/4570429. Accessed 26 Nov 2013

17. Viegas, F., Andrade, G., Almeida, J., Ferreira, R., Gonçalves, M., Ramos, G., Rocha, L.: GPU-NB: a fast CUDA-based implementation of naive bayes. In: 2013 25th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), pp. 168–175. IEEE (Oct 2013)
18. Zhou, L., Yu, Z., Lin, J., Zhu, S., Shi, W., Zhou, H., Zeng, X. et al.: Acceleration of Naive-Bayes algorithm on multicore processor for massive text classification. In: 2014 14th International Symposium on Integrated Circuits (ISIC), pp. 344–347. IEEE (Dec 2014)