EasyChair Preprint

№ 15191

Performance Analysis of Parallel Programs with RAPIDS as a Framework of Execution

Seyi Ogunji, Moises Sanchez Adame, Oscar Montiel Ross and Juan J. Tapia

October 9, 2024

Performance Analysis of Parallel Programs with RAPIDS as a Framework of Execution

Ogunji Seyi Tope[1], Moises Sanchez Adame[2], Oscar Montiel Ross[3], and Juan J. Tapia[4]

[1] Instituto Politécnico Nacional,Mexico city,CITEDI, Mexico
seyitope@citedi.mx
[2] Instituto Politécnico Nacional,Mexico city, CITEDI,Mexico
mosancheza@ipn.mx
[3] Instituto Politécnico Nacional,Mexico city, CITEDI,Mexico
oross@ipn.mx
[4] Instituto Politécnico Nacional,Mexico city, CITEDI Centre,Mexico @ipn.mx

Abstract

In this age where data is growing at an astronomical rate, with unfettered access to digital information, complexities have been introduced to scientific computations, analysis, and inferences. This is because such data could not be easily processed with traditional approaches. However, with innovative designs brought to the fore by NVIDIA and other market players in recent times, there have been productions of state-of-the-art GPUs such as NVIDIA A100 Tensor Core GPU, Tesla V100, and NVIDIA H100 that seamlessly handle complex mathematical simulations and computations, Artificial Intelligence, Machine Learning, and high-performance computing, producing highly improved speed and efficiency, with room for scalability. These innovations have made it possible to efficiently deploy many parallel programming models like shared memory, distributed memory, data parallelization, and Partitioned Global Address Space (PGAS) with high-performance metrics. In this work, we analyzed the parquet-formatted New York City yellow taxi dataset on a RAPIDS and DASK-supported distributed data-parallel training platform using NVIDIA multi-GPUs. The dataset was used to train Extreme Gradient Boosting (XGBoost), RandomForest Regressor, and Elastic Net models for trip fare predictions. Our models achieved notable performance metrics. The XGBoost achieved a mean squared error of 11.38 and R-squared of 0.9678. The model training and evaluation time took 38.51 seconds despite the huge size of the training dataset, showing how computationally efficient the system was. The model results for the RandomForest MSE was 21.96, and the R-squared was 0.9378. In the bid to show the scalability and versatility of our experimental design to different machine learning domains, our GPU-accelerated training was extended to image classification tasks by using MobileNet-V3-Large pre-trained architecture on a CIFAR-100 dataset. We achieved a ROC_AUC of over 95% for the implementation. This work advances the state-of-the-art in parallel computing through implementation of RAPIDS and DASK frameworks on a distributed data-parallel training platform making use of NVIDIA multi-GPUs. The work is built on a well established theoretical framework using Amdahl and Gustafon's laws on parallel computation.By integrating RAPIDS and DASK, we contribute to advancing parallel computing capabilities, offering potential applications in smart city development and the field of logistics and transportation management services where rapid fare predictions are very important. The contribution could also be extended to the field of image classification, vision systems, object detection and embedded systems for mobile applications.

1 Introduction

The field of scientific computing is replete with different GPU-powered parallel programming models because they are very inevitable in the efficient processing of large datasets that are available in real-world use cases. They have a mathematical framework on Amdahl and Gustafson's equations [2][4] [8] which are expressed as:

$$S(n) = \frac{1}{(1-p) + \frac{p}{n}} \tag{1}$$

and

$$S(n) = n - \alpha(n-1) \tag{2}$$

Where:

- $S(n)$ is the speedup achieved using n processors.
- p is the parallelizable portion of the program.
- $1-p$ is the sequential (non-parallelizable) portion of the program,
- n is the number of processors.
- α is the fraction of time spent on the serial (non-parallelizable) portion of the program and
- $n-1$ is the number of parallel workers relative to the serial execution.

They provide conceptual frameworks that influence the logical paths employed by programmers either during application or algorithm design or when fixing bugs in the system. The models are multifaceted: ranging from shared memory, distributed memory, hybrid model, data-parallel model, and many more. The focus of this work is on the use of RAPIDS, which is an open-source library that unlocks the speed of GPU for cost-effective computations [21]. This open-source library is based on the data parallel model. When used within a single GPU, it uses shared memory parallelism. It is built on the CUDA programming model [21]. Even though RAPIDS itself does not implement distributed memory parallelism, as shown in this work, it could be used alongside frameworks such as DASK for distributed computing across several nodes and multiple GPUs.

The remaining part of this paper is structured as follows: Section 2 describes the methodologies that were used for the experiments. Section 3 addresses the results with a detailed explanation of it. Section 4 discusses the conclusion inferred from the work and the focus for future work.

2 Methodology

This work demonstrates sophisticated pipelines used in the two experiments where the RAPIDS environment was used in conjunction with the DASK framework to enhance proper implementations of distributed data-parallel training. The work of [23] revealed the evolution of parallel computing and talked about data parallelism, model parallelism, and pipeline parallelism; we termed these DMP parallelisms. The methodology used in this work focused more on the data and pipeline parallelisms where the DASK framework was used in conjunction with RAPIDS for parallelism of data and pipelines across multiple GPUS.

Beyond the theoretical framework in the literature, we provided some empirical data to show the viability of these approaches.
Each of these experiments is analyzed below with their corresponding pipeline.

Experiment 1: Distributed Data-Parallel Training of New York City Taxi Dataset using XGBoost with RAPIDS Integration

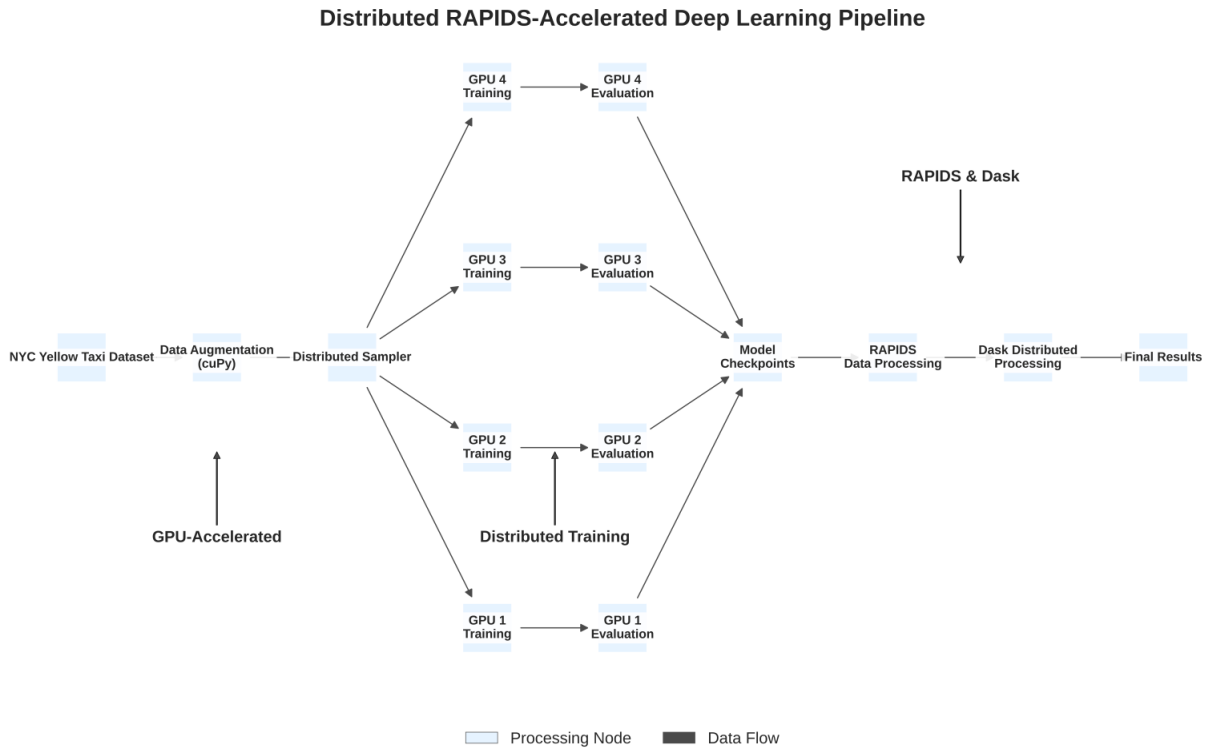


Figure 1: RAPIDS_DASK Distributed Training Pipeline for NYC Taxi Data

In the first experiment, the popular New York City yellow taxi dataset https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page, with records of 3,539,193 rows and 19 columns for June 2024, was used and trained with traditional machine learning algorithms comprising XGBoost[5],that relies on this mathematical framework:

$$\text{Obj}(\theta) = L(\theta) + \Omega(\theta) \tag{3}$$

.

Where:

- **Obj(θ)** is the objective function to minimize during training.
- **L(θ)**: The loss function, representing the error of predictions.
- $\Omega(\theta)$: Regularization term to control model complexity.

Random Forest[3],mathematically represented by the equation:

$$\hat{y} = \frac{1}{M} \sum_{m=1}^{M} f_m(x) \tag{4}$$

Where:

- $\hat{y}$ is the Predicted value.
- M is the number of decision trees in the random forest.
- $f_m(x)$ is the prediction from the m-th decision tree.

and Elastic Net[24], architecturally described by this mathematical expression:

$$L(\beta) = \|y - X\beta\|^2 + \lambda_1\|\beta\|_1 + \lambda_2\|\beta\|_2^2 \tag{5}$$

Where:

- $L(\beta)$: Elastic net loss function.
- $\|y - X\beta\|^2$ is the Residual sum of squares.
- $\lambda_1\|\beta\|_1$ is the Lasso (L1) regularization term.
- $\lambda_2\|\beta\|_2^2$ is the Ridge (L2) regularization term.

XGboost has proven to be a reliable algorithm in this type of predictive task as shown by M. Poongodi et al.[19] in their work. Even though the accuracy of their prediction was not explicitly revealed in the literature, in our case, we were able to achieve above 99% for predictions of some price ranges. The New York City yellow taxi dataset has been used widely in various machine learning applications, particularly for fare prediction and trip duration estimation like in many other works where taxi-related datasets have been used[6][14].

By taking advantage of cuPy, cuML, and cuDF libraries of the RAPIDS framework, the models were trained over a distributed data-parallel system taking advantage of the recent revolutionary trends in scalable general-purpose GPU computing [22]. The model was later used to make fare predictions. The performances of the models were tracked for visualizations.

Experiment 2: Distributed Data Parallel Training of CIFAR-100 on MobileNet-V3-Large using PyTorch and DASK with RAPIDS Integration

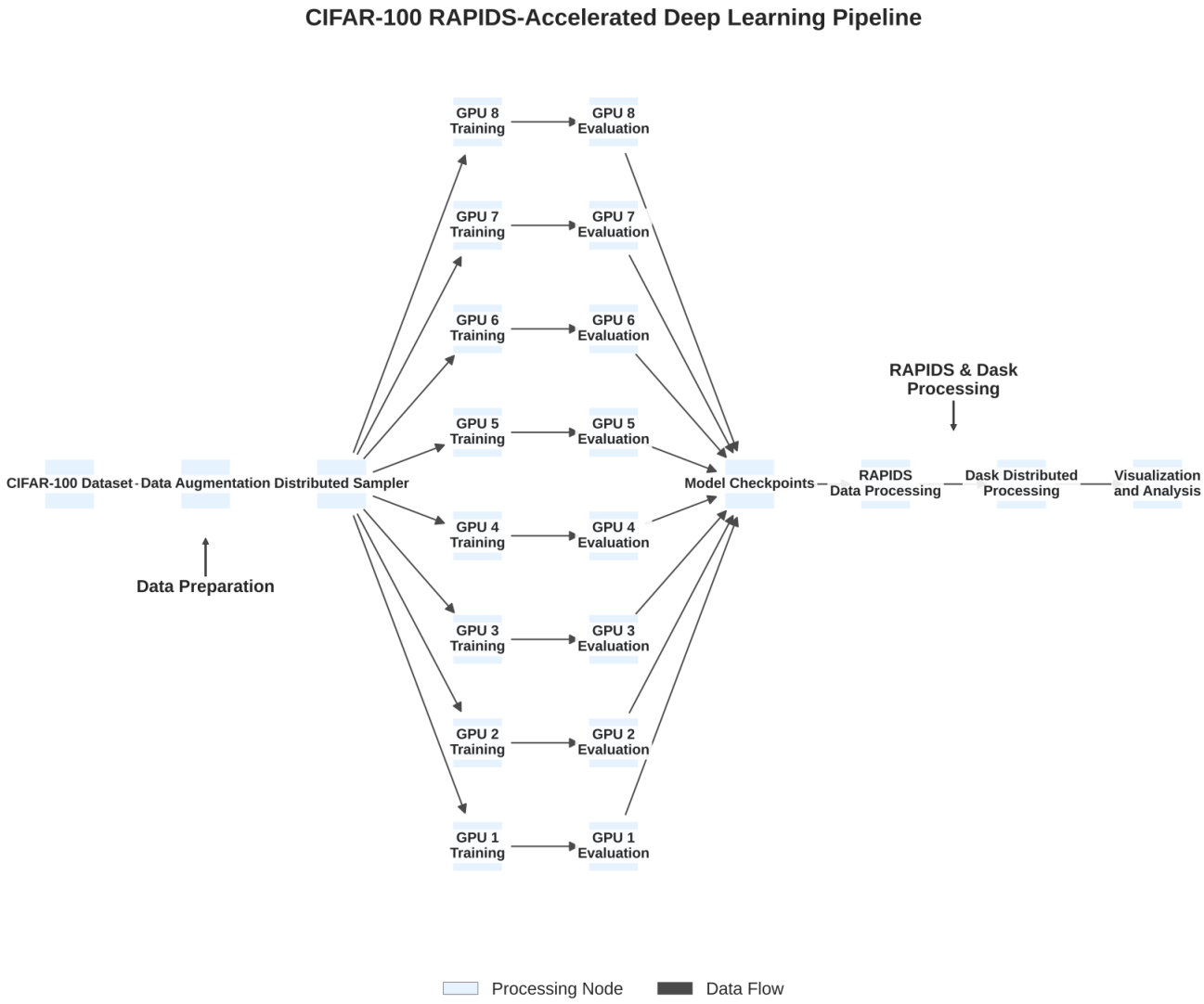


Figure 2: RAPIDS_DASK Distributed Training Pipeline

This setup utilizes some of the libraries that are provided by RAPIDS, which include cuDF for GPU-accelerated data frame manipulations and computations. It also utilizes cuPy for GPU-accelerated array operations. cuML was used as well, as the experiment is a rigorous implementation of deep learning architecture [21].

The methodology further incorporates DASK for optimum parallelism of the data across multi-GPUs, which in this experiment were eight.The speed-up gain[10] could be seen in this

expression:

$$S = \frac{T_1}{T_p} \tag{6}$$

Where:

- S is the Speedup due to parallelization.
- $T_1$is the Execution time of the sequential algorithm.
- T_p is the Execution time using p processors.

The PyTorch library was further used to complement the DASK framework for better acceleration of the task [18].

The architecture used for the experiment was a pre-trained MobileNet-V3-Large[12], which was further fine-tuned before the final classification layer with the introduction of 512 neurons and a dropout normalization strategy to enhance better model performance and generalization [11]. For the final classification into classes, the softmax[7] applied to the logits during the loss calculation is used since it has been internally incorporated with the CrossEntropyLoss function[16].

$$\sigma(z)_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)} \tag{7}$$

Where:

- $\sigma(z)_i$: Probability output for the i-th class.
- z_i: Input score for the i-th class.
- $\sum_j \exp(z_j)$: Sum of exponential scores for all classes.

$$H(p, q) = -\sum_x p(x) \log(q(x)) \tag{8}$$

Where:

- $H(p, q)$: Cross-entropy loss.
- $p(x)$: True probability distribution.
- $q(x)$: Predicted probability distribution.

A series of data augmentation techniques such as random crop, random horizontal flip, and color jitter were used on the architecture. The AdamW optimizer was used and the model was trained through 75 epochs [15].

3 Results and Discussion

Experiment 1 Results

The results obtained through the multi-GPU process using RAPIDS were very revealing. To begin with, the RAPIDS performance analysis is a great testament to how efficient and computationally fast a distributed system could be by leveraging RAPIDS' capabilities [21]. Judging from the total execution time of 39.71 seconds for a dataset of 3,539,193 records with 19 columns(Figure 4a), the enormously great benefits inherent in this setup cannot be overstated.

Also, the model training and evaluation dominated the execution time (38.51 seconds), which clearly indicates effective parallelization of the computationally intensive steps of this procedure. The timing results from the data preprocessing steps, which are splitting, scaling, and feature engineering, showed that the operations were completed in milliseconds, showcasing the power of GPU-accelerated operations.

The performance of XGBoost for trip fare prediction was superb(Figure 4b). The mean squared error [9], expressed as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{9}$$

Where:

- **MSE** stands for the Mean squared error and represents a measure of prediction accuracy.
- n is the number of data points.
- y_i is the actual value for the i-th data point.
- $\hat{y}_i$is the predicted value for the i-th data point

was 11.38, and the R^2[17], given as

$$R^2 = 1 - \frac{\text{SS}_{\text{res}}}{\text{SS}_{\text{tot}}} \tag{10}$$

Where:

- R^2 is the coefficient of determination.
- $\mathbf{SS_{res}}$ is the sum of squares of residuals.
- $\mathbf{SS_{tot}}$ is the Total sum of squares

The model's R^2 was computed as 0.9678, indicating that the model could explain nearly 97% of the variance in taxi fare data. The high accuracy (ranging from 97.63% to 99.45%) within the $1 to $10 fare prediction range suggests its real-world applicability. The top three features of importance identified by the model were trip distance, trip duration, and fare per mile.

Based on the high accuracy of the model, especially from the Mean Squared Error (MSE), we developed a **4Ps** model to illustrate its relevance. Each **P** is explained below:

Planning: The model revealed a strong correlation between trip distance, trip duration, and fare per mile. Transportation networks can utilize these features to forecast fares accurately, facilitating better scheduling and resource allocation.

Pricing: The model's accuracy ensures fair pricing for both passengers and drivers. Passengers receive reliable fare estimates, while drivers can predict their earnings on a trip. With high predictive accuracy for lower fares, such services remain accessible to lower-income earners. In conclusion, all stakeholders benefit from fair pricing.

Prevention: Transportation companies can use the model's accuracy to optimize fleet routes, helping to control fuel costs and improve service efficiency.

Prediction: In smart cities, this predictive model can aid traffic flow management and resource allocation, easing congestion issues.

Table 1: Fare Distribution Analysis by Day of the Week

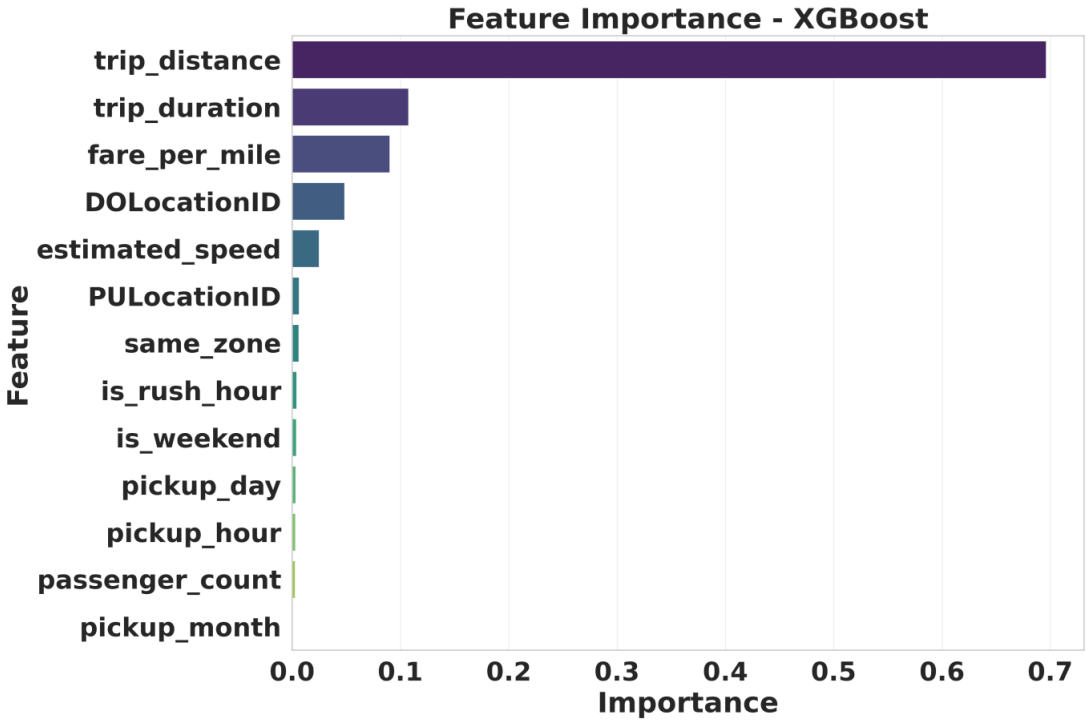
Fare Distribution Feature	**Description and Analysis**
The consistency of the Median Fare	The median fare is notably stable for all the days of the week, revealing a pattern that is uninfluenced by the erratic demand pattern that does occur at weekends and some weekdays.
Outliers in High Fare Trips	The long tails or whiskers extending upwards, particularly on certain days, reveal that high fares seldom occur. These might not be unconnected with long-distance or some special cases.
Symmetry Across Days	A unique and revealing pattern concerning the fare distributions across the week's days is noticed. A safe conclusion that could be drawn from the pattern is that day-of-week factors do not influence fare pricing tremendously.
Uncommon High-Fare Trips	The extended tops of the violins represent rarely occurring high-fare trips (above $200), stretching the fact that the model is optimized for shorter, more common trips, with less accuracy in high-fare predictions.

Table 2: Error Distribution Analysis of the XGBoost Model

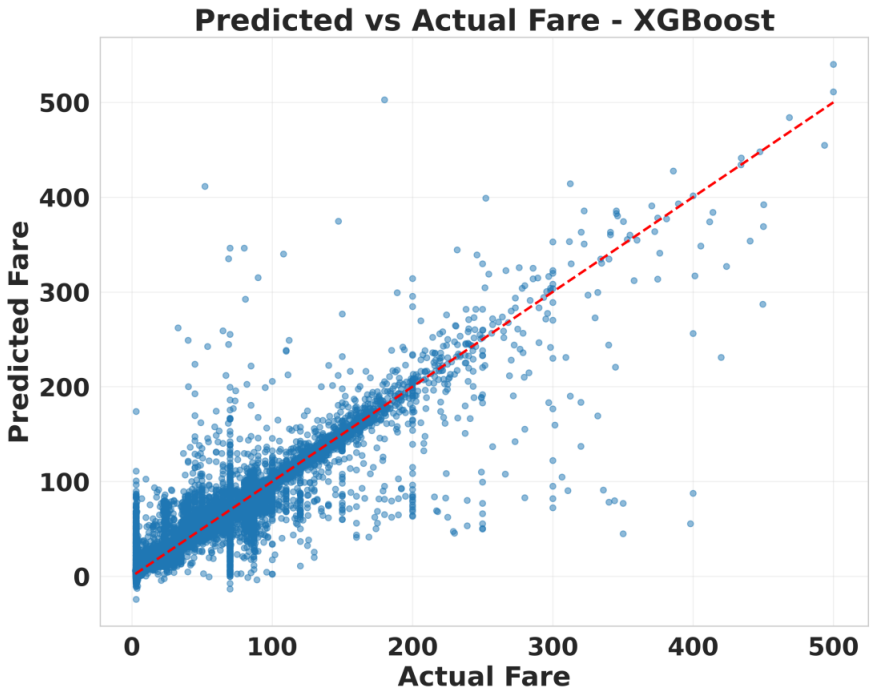
Error Distribution Feature	**Description and Analysis**
Tight Distribution Around Zero	The majority of prediction errors are concentrated around zero, indicating minimal bias and high accuracy in fare predictions for most cases.
Skewness of Errors	Thin tails are observed extending to both negative and positive sides, indicating occasional under-predictions and over-predictions, but large errors are infrequent.
Absence of Systematic Bias	The symmetric shape of the error distribution suggests that the model does not consistently overpredict or underpredict fares, enhancing its reliability.
Extreme Outliers	A few outliers with prediction errors greater than ± 300 are present, indicating that the model struggles with infrequent, high-variance fare cases, such as long trips or anomalies.

The violin plots (Figure 5b and Table 1) provided further insights into the model's performance. There were consistent median fares across all days, with fares ranging from $0 to $100. The violins' bulging at similar points suggested common fares for short trips.

These observations further confirm the model's reliability. The correlations make this model a valuable tool for transportation stakeholders in fare-related decision-making. Finally, the dense clustering of data points along the dotted diagonal line (Figure 3b) demonstrates a strong linear correlation between predicted and actual fares. The clustering also revealed near-perfect accuracy for fares in the $0 to $200 range, with increased scatter for fares above $200, indicating reduced accuracy for higher fares.

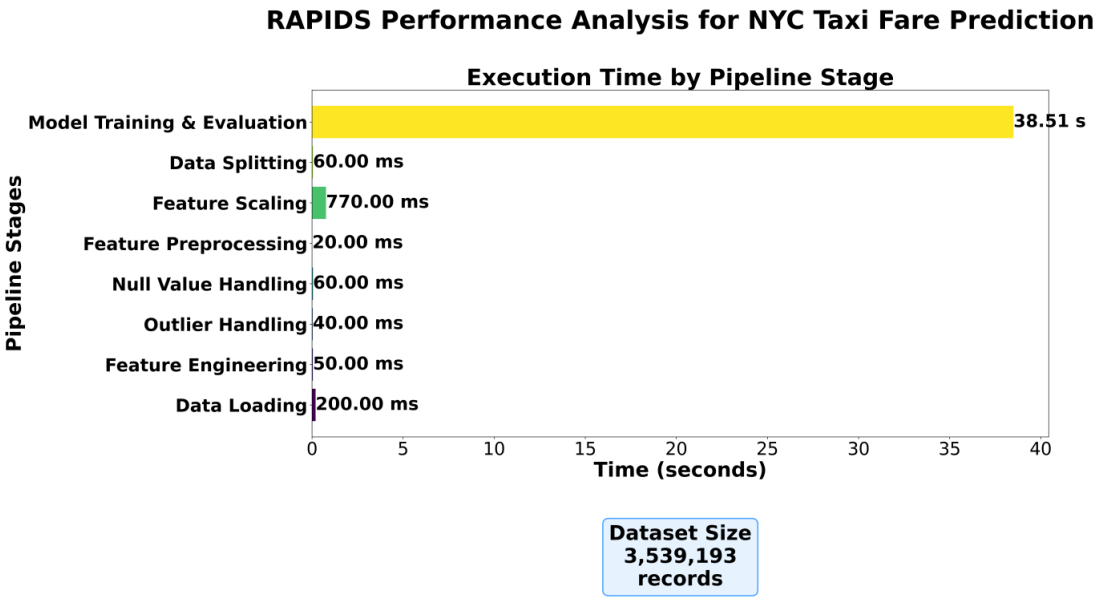


(a) Feature Importance (XGBoost)

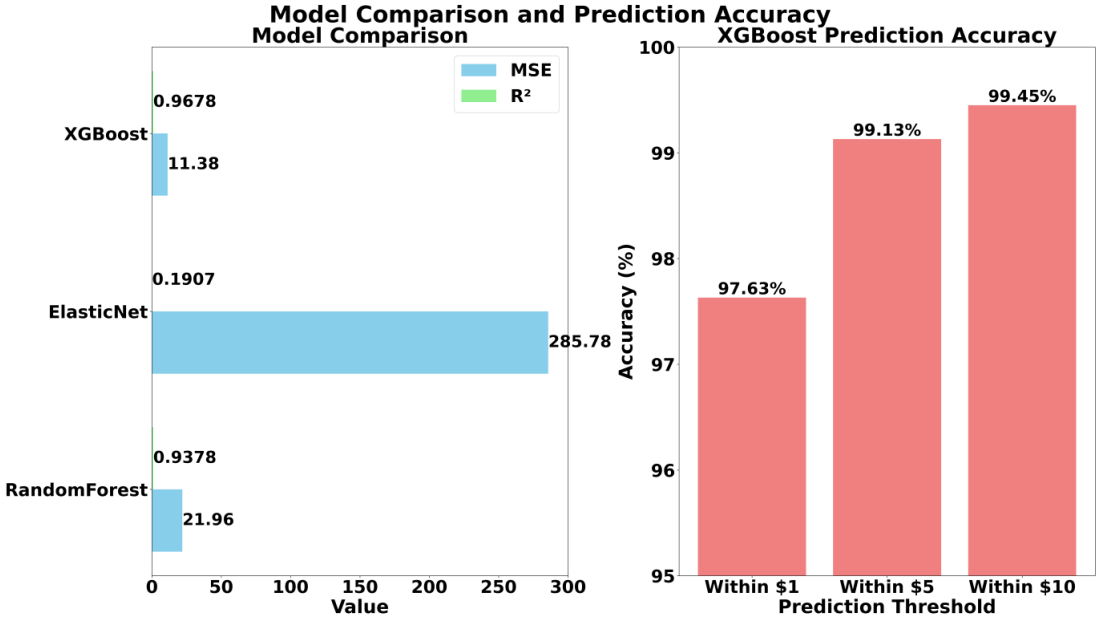


(b) Predicted vs Actual (XGBoost)

Figure 3: XGBoost Model Analysis



(a) Training Time Stages



(b) Model Comparison

Figure 4: Training Performance and Model Comparison

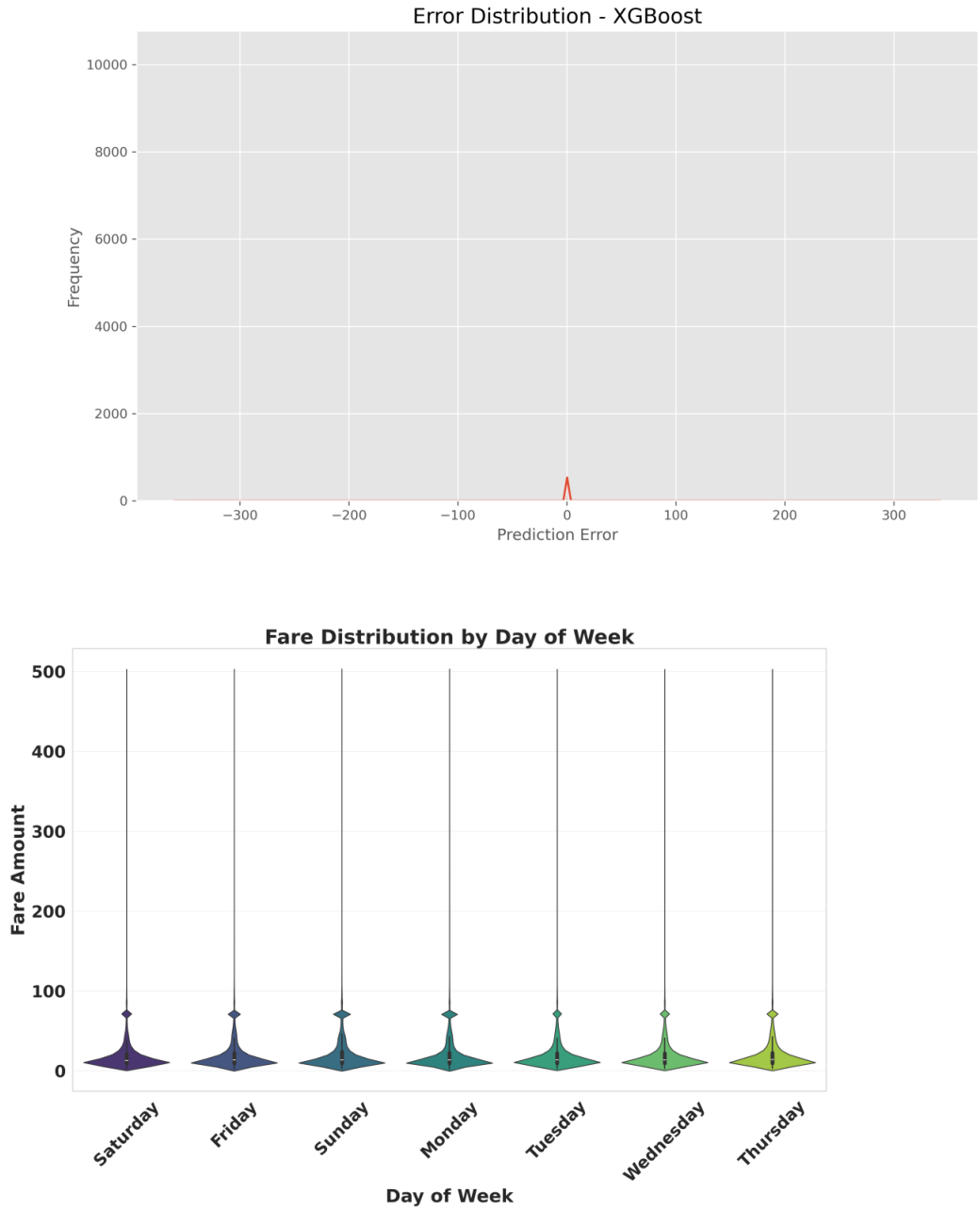


Figure 5: Error and Fare Distribution Analysis

Experiment 2 Results

For the CIFAR-100 dataset experiment, the distributed training on MobileNet-V3-Large showed promising results. The training and validation losses, Figure 10, exhibited variances across the training epochs. While the training losses dipped towards the minimum, the same was not true

for validation losses as it played more in the region between 2.5 and 3.0. These loss performances are ultimately reflected in the model's accuracy as well as shown in Figure 11.

The model's computational performance and efficiency were relatively stable throughout the training epochs as could be deduced from the execution time (figure 8) that stayed within the range of 9.5 to 12.5 seconds throughout the process. The dips and spikes notwithstanding, the GPU utilization efficiency was still relatively stable.

From Figure 9, we summarized the distribution of the training and the validation loss depicted by the violins using the table shown below.

Table 3: Comparison of Training and Validation Loss Distribution

Metric	**Training Loss**	**Validation Loss**
Range	1.5 to 4.5	2.0 to 2.5
Center	Most values centered around 2.5 to 3.5	Most values centered around 2.0 to 2.5
Shape	Wide distribution with significant fluctuations and outliers	Narrower distribution with tighter variance
Implication	Indicates the model is still learning but may be overfitting in some cases	Consistent performance on validation data, suggesting good generalization
Comparison	Wider loss distribution than validation, possible overfitting or instability	Indicates better generalization but refinement needed for training loss

The model performed well on the top 20 classes(figure 12) with predicted labels and true labels being accurately the same in most cases as shown by the confusion matrix which is mathematically represented as:

$$C_{ij} = \sum_{k=1}^{n} 1(y_k = i \wedge \hat{y}_k = j) \tag{11}$$

Where:

- C_{ij} is the Confusion matrix entry for true label i and predicted label j.
- y_k is the True label for the k-th sample.
- $\hat{y}_k$ is the Predicted label for the k-th sample[20]

The radar plot, figure 7, gave significant insights into the accuracy level of the top 10 classes, with classes 68 and 53 gravitating greatly towards the 90% accuracy point.

The Receiver Operating Curves(ROC), figure 6 gave another vivid revelation about this distributed data-parallel (DPP) training model. The Area under the Curve(ROC-AUC) for the model showed a predictive power about the model with a higher number of counts clustering around 0.9 to close to 1.0. This means that the model classifies well above the random classification.

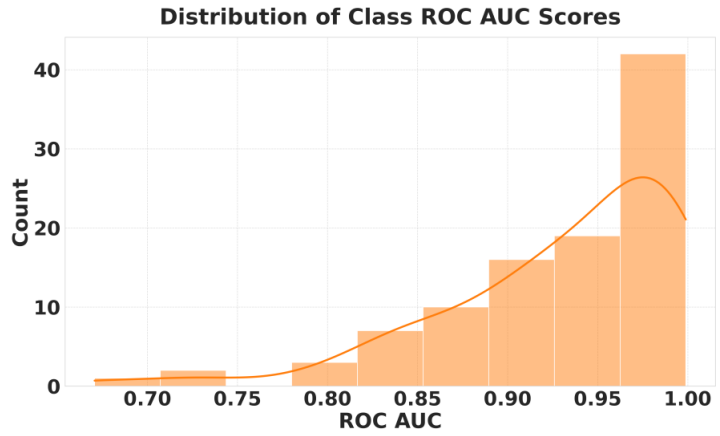


Figure 6: ROC Curves for 75 Epochs (AdamW)

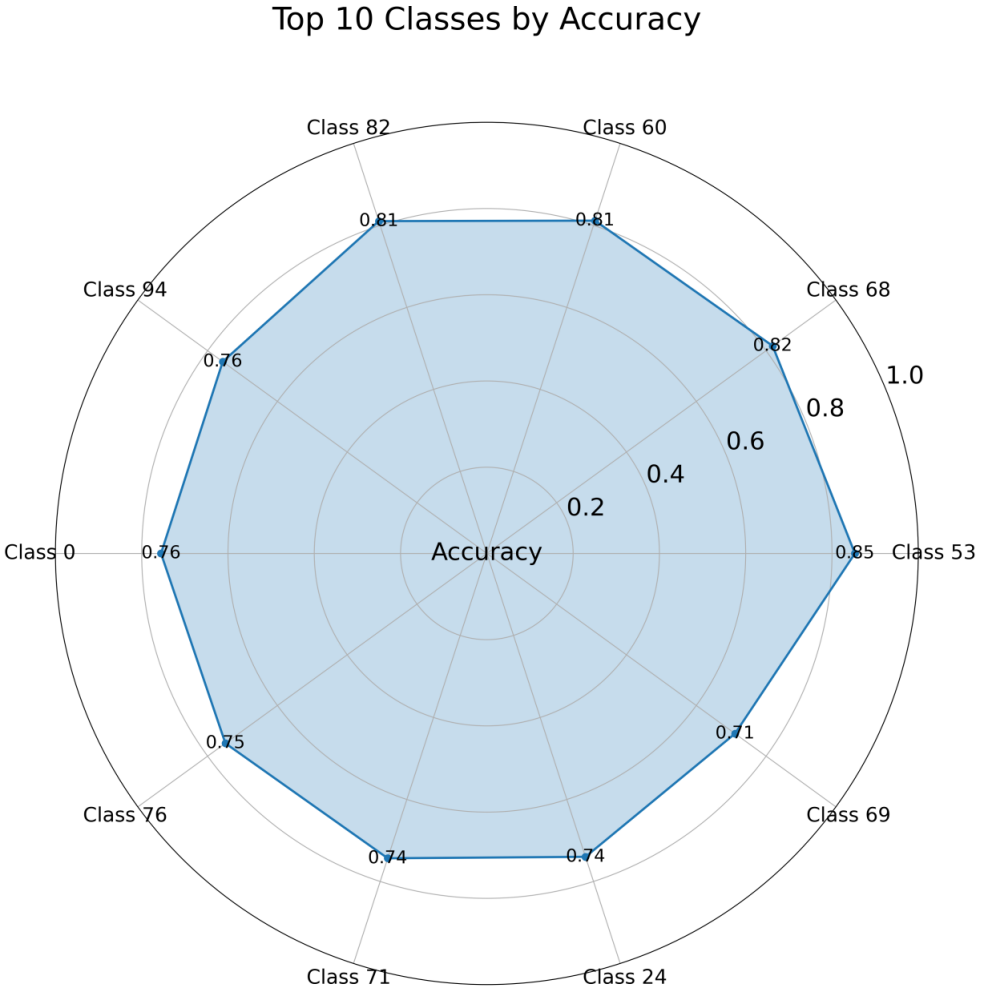


Figure 7: Radar Plot for 75 Epochs (AdamW)

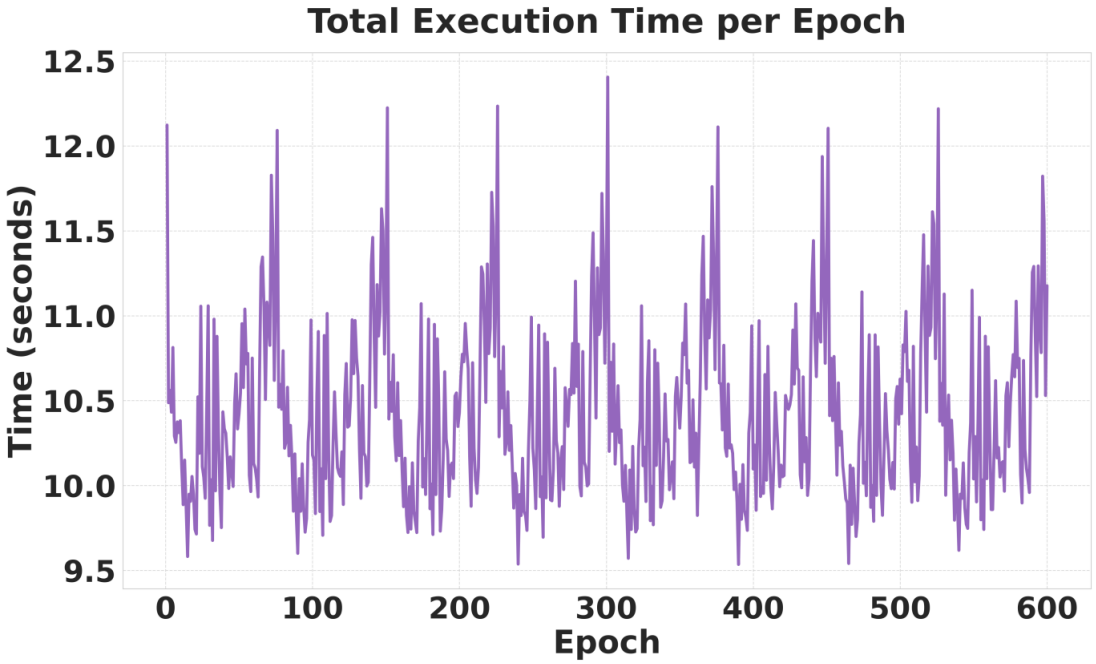


Figure 8: Training and Validation Loss for 75 Epochs (AdamW)

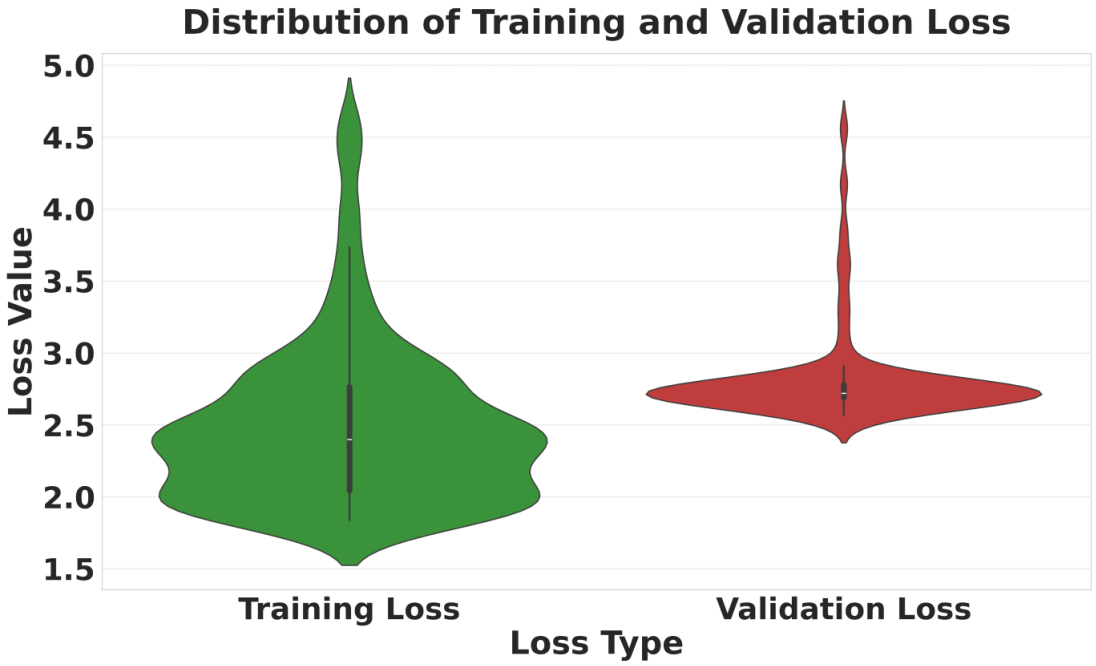


Figure 9: Training and Validation Accuracy for 75 Epochs (AdamW)

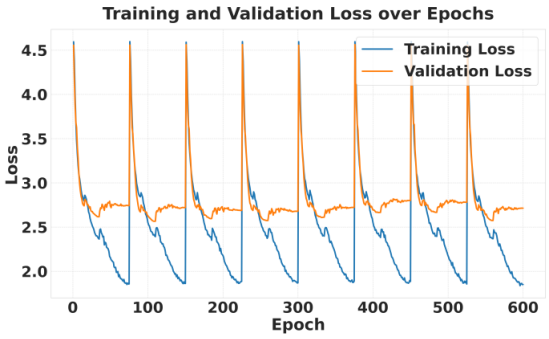


Figure 10: Training and Validation Loss

Figure 11: Training and Validation Accuracy

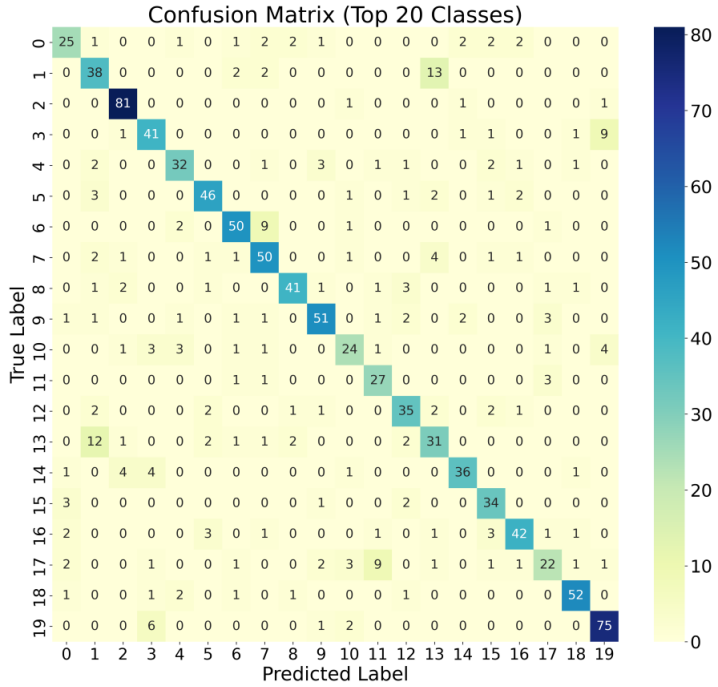


Figure 12: Confusion Matrix for 75 Epochs (AdamW)

4 Conclusion

This work has revealed a great deal of information relating to parallel programming performances in relation to efficiency, scalability, and adaptability while using RAPIDS libraries across multiple GPUs.

The first experiment has shown that with proper experimental setup and relevant model pipelines, parallel programming could be used to design scientific Models that would be robust enough to have application in the real-world dataset, and as such, informative and impactful details could be seen promptly from data points which might be very difficult to achieve when working on only CPU-based Machine. This was aptly demonstrated with the large parquet formatted dataset that was used for the trip fare prediction.

The same is equally true for the complex CIFAR-100 dataset that has applications in real-world scenarios in areas like object recognition, vision systems, image detection, mobile applications, and more. The parallelized model could be embedded in real-life gadgets for the use of humanity.

This study demonstrates the effectiveness of using RAPIDS in conjunction with DASK for distributed data-parallel training. The experiments with the NYC taxi dataset and CIFAR-100 showcased significant speedups in data processing and model training. The use of Parquet files for data storage further enhanced the efficiency of our data handling pipeline[1].

The combination of RAPIDS, DASK, and PyTorch provided a powerful framework for handling large-scale datasets and complex deep-learning models.

In our future work, we would explore how some complex datasets could be finely trained on the parallel system with a control handle on overfitting. Equally, further work would be done on much more efficient implementations of RAPIDS setup and DASK without Pytorch. This is necessary in order to discover some inefficiencies that might be introduced into the system during data transmission among the GPUS. By using the Karp-Flatt coefficient[13], we would study the inefficiencies in the system during parallelization by also focusing on factors such as communication and synchronization overheads.

Acknowledgements

The research leading to these results has received funding from the Instituto Politécnico Nacional (IPN), under Research Projects SIP20240164 and SIP20241558, and the Consejo Nacional de Humanidades Ciencias y Tecnologias (CONAHCYT) through project CF-2023-I-108.

References

[1] Daniel Abadi, Rakesh Agrawal, Anastasia Ailamaki, Magdalena Balazinska, Philip A Bernstein, Michael J Carey, Surajit Chaudhuri, Jeffrey Dean, AnHai Doan, Michael J Franklin, et al. Parquet: Efficient data storage for big data applications. In *Proceedings of the 2024 ACM SIGMOD International Conference on Management of Data*, pages 1631–1636. ACM, 2024.

[2] Gene M Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. *AFIPS spring joint computer conference*, pages 483–485, 1967.

[3] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[4] Randal E. Bryant and David O'Hallaron. *Computer Systems: A Programmer's Perspective*. Pearson Education, 3 edition, 2016.

[5] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

[6] Nivan Ferreira, Jorge Poco, Huy T. Vo, Juliana Freire, and Cláudio T. Silva. Visual exploration of big spatio-temporal urban data: A study of new york city taxi trips. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2149–2158, 2013.

[7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[8] John L Gustafson. Reevaluating amdahl's law. *Communications of the ACM*, 31(5):532–533, 1988.

[9] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2009.

[10] John L Hennessy and David A Patterson. *Computer Architecture: A Quantitative Approach*. Elsevier, 2011.

[11] Andrew Howard, Menglong Tan, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324, 2023.

[12] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. In *arXiv preprint arXiv:1704.04861*, 2017.

[13] Alan H. Karp and Harvey P. Flatt. Measuring parallel processor performance. *Communications of the ACM*, 33(5):539–543, 1990.

[14] Xiaoyuan Li, Gang Pan, Zhen Lei, and Zhenghao Huang. Spatiotemporal-aware neural networks for taxi demand prediction. *IEEE Transactions on Intelligent Transportation Systems*, 24(5):4912–4923, 2023.

[15] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2023.

[16] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT Press, 2012.

[17] Nico J D Nagelkerke. A note on a general definition of the coefficient of determination. *Biometrika*, 78(3):691–692, 1991.

[18] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, and Alban Desmaison. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 36, pages 8026–8037, 2023.

[19] M Poongodi, Mohit Malviya, Chahat Kumar, Mounir Hamdi, V Vijayakumar, Jamel Nebhen, and Hasan Alyamani. New york city taxi trip duration prediction using mlp and xgboost. *International Journal of System Assurance Engineering and Management*, pages 1–12, 2022.

[20] David Martin Powers. *Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation*, volume 2. 2011.

[21] RAPIDS AI. Rapids: Collection of libraries for end-to-end gpu data science. https://rapids.ai/, 2023. Accessed: 2024-09-15.

[22] Sebastian Raschka, Joshua Patterson, and Christoph Nolet. Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence. *Information*, 11(4):193, 2022.

[23] Shikai Wang, Haotian Zheng, Xin Wen, and Shang Fu. Distributed high-performance computing methods for accelerating deep learning training. *Journal of Knowledge Learning and Science Technology ISSN: 2959-6386 (online)*, 3(3):108–126, 2024.

[24] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.