2024 IEEE 15th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON) | 979-8-3315-4090-6/24/$31.00 ©2024 IEEE | DOI: 10.1109/UEMCON62879.2024.10754694

DyRAM: Dynamic Data Allocation and Resource Management in Distributed Machine Learning Systems

Vaibhavi Tiwari
School of Computing
Montclair State University
New Jersey, USA
tiwariv1@montclair.edu

Rahul Thakkar
School of Computing
Montclair State University
New Jersey, USA
thakkarr1@montclair.edu

Jiayin Wang
School of Computing
Montclair State University
New Jersey, USA
jiayin.wang@montclair.edu

***Abstract*—The rapid evolution of digital technologies and the pervasive nature of data connectivity have significantly expanded the scope of decentralized machine learning tasks. At the forefront of this shift is distributed machine learning, which leverages distributed data while promoting privacy and efficiency. Built on the principles of cloud computing, distributed machine learning decomposes complex computational tasks into smaller components processed concurrently across interconnected nodes, optimizing resource utilization and scalability. The global cloud computing market, integral to the advancement of distributed machine learning, is projected to grow substantially, reaching USD 2,495.2 billion by 2032. Central to this study is the Cloud-Based Ratio Proportion Data Distribution Algorithm (CB-RPDDA), an innovative solution to traditional data distribution inefficiencies. CB-RPDDA reallocates data based on the processing speeds of individual machines, ensuring optimal resource utilization and effective workload distribution. This method introduces a new perspective on dataset division among worker nodes, enhancing load balancing and performance. By integrating CB-RPDDA with distributed machine learning frameworks, we improve the efficiency of decentralized learning processes, ensuring efficient data distribution across nodes while maintaining data security and privacy. Our findings demonstrate the potential of combining CB-RPDDA with distributed machine learning to offer scalable, efficient, and secure machine learning solutions, driving significant advancements in the field.**

***Index Terms*—Distributed Machine Learning, Resource Management, Data Distribution**

I. Introduction

The landscape of distributed machine learning tasks has been greatly expanded due to the rapid advancement of digital technologies and the increasing prevalence of data connectivity[1]. Effective resource management plays a crucial role in the success of distributed machine learning systems[2]. These systems make use of decentralized data to ensure privacy and efficiency. These systems leverage the power of cloud computing[3], specifically its distributed architecture, to offer machine learning solutions that are scalable and reliable.

The foundation of distributed machine learning systems is built upon the core principles of distributed computing[4]. Within this framework, intricate computational tasks are broken down into smaller components, which are then processed simultaneously across interconnected nodes or servers. This architecture allows for efficient use of resources and scalability, effectively meeting the different needs of various machine learning workloads. Through the utilization of distributed computing, these systems offer dependable and effective services, leading to improved operational efficiency and heightened user contentment.

In addition, distributed cloud computing models are specifically designed to effortlessly handle increasing workloads by adding more nodes to the network, thereby improving processing capabilities. This ability to scale goes beyond just computational resources and includes scalable storage solutions that enhance data availability, redundancy, and resilience. In addition, distributed machine learning systems have the advantage of reducing latency by executing tasks concurrently across multiple nodes. This results in faster response times and improved overall performance[2].

In 2022, the global cloud computing market witnessed significant growth, reaching a market size of USD 495.3 billion[5]. This growth is crucial for the development of distributed machine learning. According to industry projections, there is a strong expectation for significant growth in the market. Estimates suggest that by 2032, the market could reach a value of USD 2,495.2 billion[5]. This growth is expected to be driven by a compound annual growth rate of 17.8 % from 2023 to 2032. Figure 1 provides the global statistics for cloud computing industry

This paper explores the distribution of data in distributed machine learning systems and how it affects task performance. It specifically looks at computing power of a machine to improve efficiency in cloud computing environments. Conventional methods of data distribution, such as uniform data distribution, round-robin distribution, hash-based distribution, and data sharding, typically do not consider the processing capabilities of individual nodes, leading to potential inefficiencies. In order to improve the shortcomings of this method, we present a new algorithm called the CB-RPDDA. This algorithm utilizes cloud-based ratio proportion techniques to enhance data distribution and address inefficiencies.

CB-RPDDA revolutionizes data distribution by leveraging mathematical concepts of ratio and proportion. The algorithm assigns data to machines based on their individual processing speeds, rather than distributing it uniformly. This distinctive feature guarantees efficient utilization of resources regardless of the setup, resulting in effective allocation of resources. Per-

979-8-3315-4090-6/24/$31.00 ©2024 IEEE

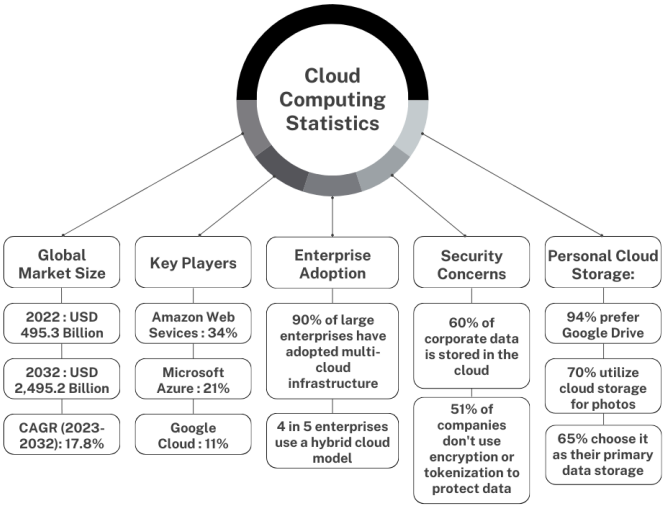

Figure 1: Statistics for the cloud computing industry

formance evaluations demonstrate that CB-RPDDA delivers significant cost reductions and efficiency enhancements when compared to conventional approaches.

Through the integration of CB-RPDDA with distributed machine learning frameworks, we are able to optimize the federated learning process by improving the efficiency of data distribution across nodes. This integration not only enhances the effectiveness of model training but also guarantees the privacy and protection of decentralized data. Ultimately, the combination of distributed machine learning and the groundbreaking CB-RPDDA algorithm offers a revolutionary method for decentralized machine learning. This integrated solution harnesses the power of distributed cloud computing to provide scalable, efficient, and secure machine learning functionalities. It has the potential to drive significant advancements in the field.

II. Related Work

In cloud computing, various data distribution techniques are employed, continually evolving to meet the demands of contemporary applications. This section provides an overview of commonly utilized strategies.

Data Partitioning and Sharding: This approach involves segmenting extensive datasets into smaller, more manageable shards, subsequently distributed across multiple nodes within the cloud infrastructure. By facilitating the simultaneous processing of different data subsets across various nodes, this strategy significantly enhances parallel processing capabilities and system scalability[6].

Replication: This technique entails creating multiple copies of data and distributing them across diverse nodes or data centers within the cloud infrastructure[7]. Replication augments data availability, fault tolerance, and resilience by ensuring data remains accessible despite potential node failures.

Consistent Hashing: This method distributes data across several nodes in a consistent manner, minimizing the need for data migration when nodes are added or removed. Consistent hashing aids in maintaining balanced workload distribution and reduces the impact of node failures on overall system performance[8].

Erasure Coding: Employed to achieve data redundancy and fault tolerance, erasure coding encodes data into a set of redundant fragments distributed across various nodes[9]. This technique facilitates data reconstruction even if some fragments are lost or corrupted, thereby enhancing data durability and resilience.

Content Delivery Networks (CDNs): CDNs distribute data to edge servers situated in different geographic regions to enhance content delivery speed and reduce latency. By caching content closer to end users, CDNs improve the performance and reliability of data-intensive applications, including web applications and streaming services[10].

Distributed Stream Processing: This innovative approach utilizes server-less functions to efficiently manage data processing tasks within a distributed stream processing framework[11]. Stream processing enables real-time data analysis as data is generated, with server-less functions providing a cost-effective and scalable solution for managing substantial and continuous data streams in cloud environments.

Blockchain-based Data Distribution: Leveraging blockchain technology, this method ensures secure and immutable data distribution among various cloud providers or organizations. Blockchain establishes an unalterable record of data transactions, ensuring data integrity and enabling secure collaboration in multi-cloud environments[12].

Fusion of Edge Computing and Fog Data Distribution: This approach amalgamates edge computing, which processes data near its source, with fog computing, which facilitates distributed processing at the network edge. By distributing data among edge and fog nodes[13], this method supports immediate analysis and decision-making based on locally sourced data, benefiting applications such as IoT data processing and real-time autonomous systems.

Optimizing Data Placement with Machine Learning: Machine learning algorithms are employed to enhance data placement across cloud storage resources. By analyzing factors such as access patterns, data size, and processing requirements, ML algorithms can recommend optimal storage locations for different datasets, leading to improved performance and cost efficiency[14].

III. Problem Formulation

In this paper, we consider a heterogeneous cluster of distributed machine learning system that consists of a resource manager, acted as a head for global data scheduling, and worker nodes, with different performance to execute the training process of the given model and data segments. Table I lists the notations we use in the rest of this paper.

We consider a set of m jobs, $\{J_1, J_2, ..., J_m\}$ are submitted to the distributed machine learning system. Assume there are k worker nodes in the cluster, $\{n_1, n_2, ..., n_k\}$. D_i is the size of the dataset for J_i.

In the cluster, C_j represents the storage capacity of the worker node N_j. Meanwhile, the performance of the training process in each worker node could change dynamically based

on various factors, such as the current resource capacity, thermal throttling of CPUs/GPUs, power supply issues, etc. For a worker node $S_j \in S$, $S[j,t]$ represents the speed for the training process of server n_j at time t.

During the training process, each job can divide its dataset and assign partial to each worker node for parallel execution. For a job $J_i \in J$, R_{ij} represents the ratio of data in J_i allocated to the worker node n_j.

Given the storage capacity of C and the model training speed S, our objective is to assign appropriate ratio of dataset to the worker nodes in the cluster of distributed machine learning system for each job, to minimize the overall execution times of all the jobs (i.e., minimize the makespan). Specifically, let st_i and μ_i be the starting time and execution time of job J_i. Our scheduling problem is to derive st_i, R_{ij} in order to

$$minimize : max\{st_i + \mu_i\}, \forall i \in J; where$$
$$\mu_i = \frac{R_{ij} D_i}{S[j, st_i]}, st_i, \mu_i \in \sigma$$

$$\sum_{j=1}^{k} R_{ij} = 1, \forall i \in J; \quad (1)$$

$$\sum_{i=1}^{m} R_{ij} D_i \leq C_j, \forall j \in N \quad (2)$$

$$R_i j \in \{0,1\}, st_i \geq 0, \forall i, \sigma > 0$$

We consider time cost is measured as discrete value which is multiple of the time units. σ here represents a particular time unit. We assume the processing speed of S_j is consistent during the same time unit. Therefore, constrain (1) indicates that the data set could be divided to 1 - k worker nodes. Constrain (2) specifies that the data allocation to each node cannot exceed its storage capacity. Assume μ_i is available and each job is independent, our scheduling problem is equivalent to general resource constrained scheduling problem which has been proven to be NP-complete[15]. In this paper, we propose a practical heuristic algorithm CB-PRDDA that can dynamically adjust the data allocation based on the changing the performance of each worker node.

Table I: Notation Table

J/N	set of jobs/ set of worker nodes in the cluster
D	The size of entire dataset in the cluster to be distributed
S_i	Processing speed of node n_i
S_{max}	Maximum processing speed among all nodes
R_i	Ratio of the processing speed of node n_i
D_i	Segment of the dataset allocated to node n_i.

IV. The Proposed Framework

In this section, we provide a comprehensive overview of the Ratio Proportion-Based Algorithm (CB-RPDDA) and its effectiveness in data distribution through simulation. We then introduce our innovative model that leverages a cloud platform, highlighting its ability to optimize resource utilization through static and dynamic allocation methods. Additionally, we discuss the static and dynamic algorithms that form the core of our model, illustrating their roles in enhancing performance and efficiency in distributed computing environments.

A. CB-RPDDA Overview

The Ratio Proportion-Based Algorithm (CB-RPDDA) introduces a novel approach to distributing datasets among multiple worker nodes by considering their varying execution speeds. This innovative method starts by executing a sample dataset on each worker node to accurately measure their processing speeds. These speeds are then meticulously recorded and organized in ascending order to identify the node with the highest performance, ensuring a clear understanding of the relative capabilities of each node. Following this, the algorithm calculates the ratio of each node's execution speed relative to the fastest node. This step is crucial as it establishes a proportional relationship between the nodes based on their processing speeds. Utilizing these calculated ratios, the dataset is divided into segments, with faster nodes receiving a larger proportion of the data, thereby optimizing resource utilization. By aligning the workload with the processing capabilities of each node, the algorithm ensures that each node operates at its optimal capacity. This approach is particularly advantageous in distributed computing systems with heterogeneous processing capacities, as it facilitates more effective load balancing. The result is a significant improvement in overall system performance and resource utilization, making the CB-RPDDA an essential tool for managing distributed datasets efficiently.

B. Static Allocation

During the static phase of the algorithm, the distribution of data is meticulously determined by assessing the processing speeds of the worker nodes. The first step in this process involves running a sample dataset on all nodes to measure their execution speeds. The speeds are ranked in ascending order, and the ratio of each node's speed compared to the fastest node is calculated. Using these ratios, the dataset is segmented, allocating larger portions of the data to the faster nodes. This approach guarantees that the distribution of tasks is in line with the processing capacity of each node, maximizing the utilization of resources right from the beginning.

This approach works best in environments where the nodes' processing capabilities remain consistent over time. Through the utilization of established performance metrics for each node, the algorithm guarantees an equitable distribution of the workload, reducing the likelihood of bottlenecks and improving the efficiency of the entire system. The initial static allocation establishes a solid groundwork for ensuring peak performance and efficient use of resources as the data processing tasks advance.

C. Static CB-RPDDA Algorithm

The static allocation ensures optimal resource utilization by aligning the initial workload with the processing capabilities of each node. This approach is particularly effective

in environments where the processing capabilities of nodes remain consistent over time, providing a balanced workload distribution based on the initial performance metrics.

In the static part of the CB-RPDDA, the initial data distribution is carried out while taking into consideration the speeds at which the executions were recorded.

This is how the steps are executed:

Algorithm 1 Static Allocation in CB-RPDDA

Require: D: Dataset, $N = \{n_1, n_2, \ldots, n_k\}$: Set of k worker nodes

Ensure: Optimally distributed dataset segments D_n for each node $n \in N$

1: **E**xecute a sample dataset on each node $n_i \in N$ and record their processing speeds S_i

$$S_i = \text{processing speed of node } n_i$$

2: **S**ort the nodes N in ascending order based on S_i

$$S_1 \leq S_2 \leq \ldots \leq S_k$$

3: **I**dentify the node n_{max} with the highest processing speed S_{max}

$$S_{max} = \max\{S_1, S_2, \ldots, S_k\}$$

4: **C**ompute the ratio R_i of each node's speed S_i to the fastest node's speed S_{max}

$$R_i = \frac{S_i}{S_{max}}, \quad \forall n_i \in N$$

5: **D**ivide the dataset D into segments D_i based on the ratios R_i

$$D_i = R_i \times D, \quad \forall n_i \in N$$

6: **A**llocate segment D_i to node n_i

$$n_i \leftarrow D_i, \quad \forall n_i \in N$$

7: **Output:** Distributed dataset segments D_i

D. Dynamic Allocation

The dynamic part of the CB-RPDDA introduces real-time adjustments to the data distribution based on continuous monitoring of the nodes' performance. Following the initial static allocation, the algorithm continuously monitors the execution speeds of the worker nodes. If a node's performance improves or declines significantly, the algorithm recalculates the execution speed ratios and reallocates the dataset segments accordingly. This real-time adjustment ensures that the data distribution remains optimal, accommodating any changes in the nodes' processing capabilities. By dynamically adjusting the workload, the algorithm enhances overall system efficiency and resource utilization, making it particularly beneficial for distributed computing systems with heterogeneous and fluctuating node performance. This dynamic approach ensures effective load balancing and maintains high performance even in environments with variable node capabilities.

E. Dynamic CB-RPDDA Algorithm

The dynamic part of the CB-RPDDA algorithm introduces real-time adjustments to the data distribution based on continuous monitoring of the nodes' performance. The steps of the dynamic allocation are as follows:

Algorithm 2 Dynamic Allocation in CB-RPDDA

Require: D: Dataset, $N = \{n_1, n_2, \ldots, n_k\}$: Set of k worker nodes

Ensure: Optimally distributed dataset segments D_n for each node $n \in N$

1: **E**xecute a sample dataset on each node $n_i \in N$ and record their processing speeds S_i
2: **S**ort the nodes N in ascending order based on S_i
3: **I**dentify the node n_{max} with the highest processing speed S_{max}
4: **C**ompute the ratio R_i of each node's speed S_i to the fastest node's speed S_{max}
5: **D**ivide the dataset D into segments D_i based on the ratios R_i
6: **A**llocate segment D_i to node n_i
7: **C**ontinuously monitor the processing speeds S_i of each node n_i
8: **while** system is running **do**
9: **if** there is a significant change in S_i **then**
10: Recompute the ratios R_i
11: Reallocate the dataset segments D_i based on the new ratios R_i
12: Allocate the new segments D_i to nodes n_i
13: **end if**
14: **end while**
15: **Output:** Continuously optimized dataset segments D_i

The dynamic allocation enhances overall system efficiency and resource utilization by continuously monitoring and adjusting the data distribution in real-time based on the nodes' processing capabilities. This approach is particularly beneficial in environments with fluctuating node performance, ensuring effective load balancing and high performance.

V. Motivation

The motivation for this research paper arises from the critical need to enhance the efficiency and performance of distributed machine learning systems within heterogeneous computing environments. Traditional data distribution methods, such as equal split strategies, often fail to leverage the diverse processing capabilities of different nodes, resulting in inefficiencies and suboptimal resource utilization. As the volume of data and the complexity of machine learning models continue to grow, these inefficiencies become increasingly problematic.

To address these challenges, this paper introduces the Cloud-Based Ratio Proportion Data Distribution Algorithm (CB-RPDDA). By dynamically allocating data based on the processing speeds of individual nodes, CB-RPDDA aims to optimize resource utilization and improve load balancing,

thereby enhancing overall system performance. The primary motivation for this study is to demonstrate that an adaptive data distribution strategy can significantly reduce processing times and increase efficiency without compromising the accuracy of machine learning models. Moreover, by improving efficiency and resource utilization, CB-RPDDA can help save both money and effort, making it a highly cost-effective solution.

Through rigorous experimentation using the Google Cloud Platform, alongside simulations conducted in Google Colab, this research seeks to provide compelling evidence of the advantages of the CB-RPDDA approach over traditional data distribution methods. The study highlights the potential improvements in efficiency and accuracy, contributing valuable insights to the field of distributed machine learning and promoting the adoption of more effective data distribution strategies in real-world applications.

VI. Performance Evaluation

A. Simulation Analysis

This section presents a comprehensive simulation study conducted on Google Colab to evaluate the efficiency and accuracy of distributed machine learning using two distinct data distribution strategies: equal split and CB-RPDDA based split. The MNIST dataset was employed in this simulation, and TensorFlow was utilized for building and training the machine learning models.

1) Methodology:

a) Equal Split Strategy: In this approach, the dataset was divided equally among five worker nodes, with each node processing an identical number of records irrespective of their processing speed. The performance metrics for this strategy were derived by assessing the average processing time, accuracy, and efficiency of each node.

b) CB-RPDDA Split Strategy: This approach allocated data to the worker nodes based on their processing speeds. Nodes with higher speeds processed a proportionately larger share of the dataset. This strategy aimed to optimize the overall efficiency of the distributed system by leveraging the nodes' processing capabilities.

2) Benchmarks and Metrics: The MNIST [16] database of handwritten digits, available from various sources, has been used as the benchmark for this step. The MNIST dataset includes a training set of 60,000 examples and a test set of 10,000 examples. The digits have been size-normalized and centered in a fixed-size image. It is an excellent database for testing learning techniques and pattern recognition methods on real-world data with minimal preprocessing and formatting effort.The Table II provides a summary of the key components of the MNIST dataset, including the training and test set images and labels along with their respective file sizes.

Table II: MNIST Dataset File Statistics

File Name	Description	Size (bytes)
train-images-idx3-ubyte.gz	Training set images	9,912,422
train-labels-idx1-ubyte.gz	Training set labels	28,881
t10k-images-idx3-ubyte.gz	Test set images	1,648,877
t10k-labels-idx1-ubyte.gz	Test set labels	4,542

To systematically compare the two data distribution strategies, the following metrics were utilized:

- **Processing Time**: The average time taken by each worker node to complete the training process.
- **Accuracy**: The performance of the machine learning model measured on the test data.
- **Efficiency**: The ratio of the total data processed to the total processing time, providing a measure of resource utilization.

3) Results: The simulation results for the equal split and Cloud-Based Ratio Proportion Data Distribution Algorithm (CB-RPDDA) split strategies demonstrate notable differences in performance metrics, as illustrated in Table III. The CB-RPDDA split strategy significantly outperformed the equal split strategy in terms of average processing time and overall efficiency. Specifically, the CB-RPDDA split achieved an average processing time of 9.76 seconds compared to 13.82 seconds for the equal split. This reduction in processing time highlights the effectiveness of CB-RPDDA in leveraging node processing speeds for optimized data distribution.

Table III: Simulation Results for Equal Split and CB-RPDDA Split Strategies

Metric	Equal Split	CB-RPDDA Split
Average Processing Time	13.82 seconds	9.76 seconds
Overall Efficiency	868.57	1230.06
Overall Accuracy	0.9077	0.9103

By comparing the processing times, we observe a percentage reduction of approximately 29.36% in favor of the CB-RPDDA split strategy. This substantial decrease underscores the efficiency gains achieved through the adaptive data distribution method. Furthermore, the overall efficiency of the CB-RPDDA split was markedly higher at 1230.06, compared to 868.57 for the equal split. This represents an approximate efficiency gain of 41.73%, indicating superior resource utilization and load balancing capabilities inherent in the CB-RPDDA approach.

Interestingly, both strategies achieved comparable levels of accuracy, with the CB-RPDDA split achieving a slightly higher overall accuracy of 0.9103 compared to 0.9077 for the equal split. This marginal increase in accuracy suggests that the CB-RPDDA not only enhances efficiency and reduces processing time but also maintains, if not improves, the predictive performance of the machine learning model. These findings underscore the potential of the CB-RPDDA split strategy to enhance the efficiency and performance of distributed machine learning systems without compromising accuracy.

B. Experimental Analysis

The implementation of CB-RPDDA was conducted on the Google Cloud Platform (GCP) to harness its robust infrastructure and scalable resources. GCP provides a flexible and powerful environment for deploying and managing distributed machine learning workloads. This section details the architecture used for the experiments, including the configuration of virtual machines, data storage solutions, and the orchestration of machine learning tasks across multiple nodes.

1) System Setup: The architecture leverages Google Compute Engine instances configured to simulate a heterogeneous computing environment with varying processing capabilities. Data was stored in Google Cloud Storage, ensuring efficient access and retrieval during the experiments. The machine learning tasks were managed using TensorFlow, with the CB-RPDDA algorithm dynamically distributing the data based on the processing speeds of each node.

To systematically configure and implement the experimental setup, we followed these steps:

a) Configuration Steps:

1) Set up the Kubernetes cluster on GCP.
2) Configure the node pool according to the specifications detailed in Table IV.
3) Initially, split records equally and distribute them across all worker nodes.
4) Utilize Docker and YAML to create the model image and define the necessary dependencies.
5) Deploy the model to the worker nodes and observe the performance metrics.
6) Execute the CB-RPDDA algorithm to dynamically split the data based on the computing capacity of each node.
7) Assign data to worker nodes proportionally according to their respective computing capacities.
8) Redeploy the reconfigured model to the worker nodes.
9) Record and analyze the performance results of the model when using the CB-RPDDA algorithm.

This comprehensive setup and execution process ensured that the CB-RPDDA algorithm was thoroughly tested within a controlled and scalable cloud environment, allowing for precise measurement of its impact on system performance and resource utilization.

Table IV: Description of Kubernetes Clusters

Cluster	CPUs	GPUs
Cluster-1	3	2 (pool-1: 2 CPU, pool-2: 1 CPU, pool-3: 2 GPU)
Cluster-2	3	2 (pool-1: 3 CPU, pool-2: 1 GPU, pool-3: 1 GPU)

b) Tools Used: To effectively implement and evaluate the CB-RPDDA algorithm within a distributed machine learning framework on Google Cloud Platform, several tools were utilized. Each tool played a crucial role in setting up the environment, managing the workflow, and conducting the experiments. Below is a brief overview of the tools used, highlighting their specific contributions to the project.

- **Google Cloud Platform (GCP)**: GCP offers a comprehensive suite of cloud computing services, facilitating scalable and efficient cloud solutions for computing, data storage, data analytics, and machine learning[17].
- **Kubernetes**: An open-source platform for automating the deployment, scaling, and management of containerized applications, ensuring efficient and reliable operations across a cluster[18].
- **Docker**: A platform for developing, shipping, and running applications inside containers, providing consistency across development, testing, and production environments[19].
- **YAML (YAML Ain't Markup Language)**: A human-readable data serialization standard used primarily for configuration files and data exchange between different languages, integral to defining container configurations in Kubernetes[20].
- **TensorFlow**: An open-source machine learning library developed by Google, utilized for various applications including neural networks, natural language processing, and image recognition, supporting both high-level and low-level APIs for model prototyping and optimization[21].
- **PyTorch**: An open-source machine learning library developed by Facebook's AI Research lab, offering a flexible platform for building deep learning models with dynamic computation graphs and seamless integration with Python[22].

2) Benchmarks and Metrics: This research employed the ImageNet and IMDb datasets as benchmarks to evaluate the performance of the CB-RPDDA algorithm on the Google Cloud Platform (GCP).

ImageNet is a comprehensive image database organized according to the WordNet hierarchy, comprising 14,197,122 images. ImageNet's extensive and structured nature makes it an ideal benchmark for assessing the efficacy of adaptive data distribution methods in image recognition and classification tasks, providing a robust basis for performance evaluation[23].

The IMDb dataset, frequently utilized for natural language processing tasks, encompasses a vast collection of movie reviews along with associated metadata such as ratings, genres, and plot summaries. This dataset is crucial for evaluating models designed for sentiment analysis, text classification, and other NLP tasks, offering a rich source of diverse textual data for thorough testing and validation[24].

To systematically compare the two data distribution strategies, the following metrics were utilized:

- **Processing Time**: The average duration taken by each worker node to complete the training process.
- **Accuracy**: The performance measure of the machine learning model on the test dataset.

3) Results: This section presents the experimental results, emphasizing the performance differences between the Equal Split and CB-RPDDA Split strategies using the IMDb and ImageNet datasets.

Table V: Experimental Results for Equal Split and CB-RPDDA Split Strategies (IMDb)

Metric	Equal Split	CB-RPDDA Split
Average Processing Time	147.56 seconds	120.11 seconds
Overall Accuracy	0.9277	0.9403

Table VI: Experimental Results for Equal Split and CB-RPDDA Split Strategies (ImageNet)

Metric	Equal Split	CB-RPDDA Split
Average Processing Time	140.16 seconds	115.56 seconds
Overall Accuracy	0.9143	0.9486

Table V and VI provide a comprehensive summary of the experimental outcomes for both datasets. For the IMDb

dataset, the CB-RPDDA Split strategy significantly reduced the average processing time from 147.56 seconds to 120.11 seconds, representing a 18.60% decrease, while also achieving a higher overall accuracy of 0.9403 compared to 0.9277 for the Equal Split strategy, indicating a 1.35% improvement. Similarly, for the ImageNet dataset, the CB-RPDDA Split strategy decreased the average processing time from 140.16 seconds to 115.56 seconds, representing a 17.56% reduction, and improved overall accuracy from 0.9143 to 0.9486, indicating a 3.75% enhancement. These results, as illustrated in Figure 2, underscore the effectiveness of the CB-RPDDA algorithm in optimizing processing efficiency and enhancing model accuracy.

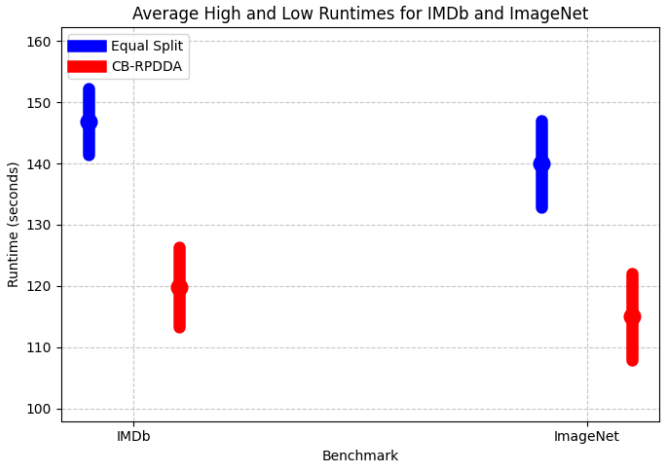

Figure 2: Runtime comparison for Equal Split and CB-RPDDA Split strategies (IMDb and ImageNet)

C. Discussions

An evaluation of the CB-RPDDA algorithm revealed several critical advantages in the context of distributed machine learning systems. By initially sending mock or split data, CB-RPDDA can ensure that all nodes are operational before proceeding with full data disclosure, thereby enhancing security against potential cyber attacks such as ransomware[25]. This incremental data distribution approach mitigates the risk of a complete data breach by not exposing the entire dataset simultaneously. Moreover, even if one node is compromised, the full dataset remains protected. The dynamic aspect of CB-RPDDA makes it particularly effective for both streaming and batch data processing. As data is continuously ingested, the algorithm can adjust data allocations in real-time based on node performance metrics, ensuring optimal load balancing and resource utilization. This flexibility is especially beneficial in heterogeneous computing environments, where nodes have varying processing capabilities. By dynamically distributing data according to these capabilities, CB-RPDDA significantly improves processing efficiency and overall system performance, underscoring the algorithm's robustness and adaptability in diverse operational scenarios.

VII. Conclusion

In recent years, the landscape of distributed machine learning has expanded significantly, driven by advancements in digital technologies and the increasing connectivity of data. Efficient resource management has become critical for the success of distributed machine learning systems. In this paper, we introduced the Cloud-Based Ratio Proportion Data Distribution Algorithm (CB-RPDDA), a novel framework designed to optimize data distribution across heterogeneous worker nodes. Through comprehensive simulations and actual experiments conducted on the Google Cloud Platform, we compared the CB-RPDDA split strategy against the traditional equal split strategy using the MNIST, IMDb, and ImageNet datasets.

Our results demonstrated that the CB-RPDDA split strategy significantly outperforms the equal split strategy in terms of average processing time and overall efficiency. For the MNIST dataset, tested through simulation, CB-RPDDA achieved a 29.36% reduction in processing time and a 41.73% increase in efficiency. Similarly, for the IMDb dataset, tested in a GCP-based controlled environment, the CB-RPDDA split reduced the average processing time by 18.61% (from 147.56 seconds to 120.11 seconds) and improved overall accuracy by 1.35% (from 0.9277 to 0.9403). For the ImageNet dataset, also tested in a GCP-based controlled environment, the CB-RPDDA split decreased the average processing time by 17.57% (from 140.16 seconds to 115.56 seconds) and increased overall accuracy by 3.75% (from 0.9143 to 0.9486).

These improvements underscore the effectiveness of CB-RPDDA in leveraging node processing speeds for optimized data distribution, leading to enhanced resource utilization and load balancing. These findings highlight the potential of the CB-RPDDA approach to enhance the performance and efficiency of distributed machine learning systems without compromising accuracy. Future work will focus on extending the CB-RPDDA framework to other machine learning tasks and evaluating its scalability and robustness in more complex distributed environments.

VIII. Future Scope

The Cloud-Based Ratio Proportion Data Distribution Algorithm (CB-RPDDA) demonstrates significant potential for further enhancing its utility and adaptability within distributed machine learning environments. Several promising avenues can be explored to advance its capabilities. One key area for future research is the algorithm's application across a broader range of distributed systems, particularly those with varying computing resources and dynamic workloads. As these systems evolve, becoming more diverse and complex, CB-RPDDA could be refined to effectively manage scenarios involving different network latencies, fluctuating node availabilities, and diverse data processing needs. Expanding this project would require further refinement of the algorithm to account for factors such as network bandwidth and node-specific constraints, thereby optimizing performance across a wider array of system architectures.

Additionally, there is significant potential in integrating CB-RPDDA with emerging technologies like edge computing and

fog computing, which enable data distribution and processing closer to the data source. By adapting the algorithm for efficient data management in decentralized environments, it could be effectively applied to Internet of Things (IoT) applications and real-time analytics. Incorporating machine learning models into CB-RPDDA could further enhance its ability to predict node performance and make real-time adjustments to data allocation, leading to more efficient resource utilization. Furthermore, future work will involve benchmarking CB-RPDDA against other existing data distribution techniques, beyond the equal split method currently employed. Testing the algorithm against methods such as hash-based distribution, consistent hashing, and replication strategies will provide a more comprehensive understanding of its relative strengths and weaknesses. As distributed machine learning systems continue to expand in scale and complexity, these advancements will be crucial in establishing CB-RPDDA as a state-of-the-art solution for streamlined data management and resource optimization.

IX. Acknowledgment

The authors gratefully acknowledge the support of the US National Science Foundation, provided through the MRI grant for the Acquisition of a High-Performance GPU Cluster for Research and Education (CNS-2018575). This funding has significantly contributed to the advancement of the research and educational initiatives described in this paper.

References

[1] L. Belcastro, F. Marozzo, and D. Talia, "Programming models and systems for big data analysis," *International Journal of Parallel Emergent and Distributed Systems*, vol. 34, pp. 632–652, 2019.

[2] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, "A survey on distributed machine learning," *ACM Computing Surveys*, vol. 53, no. 2, pp. 30:1–30:33, 2020. [Online]. Available: https://doi.org/10.1145/3377454

[3] E. J. Qaisar, "Introduction to cloud computing for developers: Key concepts, the players and their offerings," in *2012 IEEE TCF Information Technology Professional Conference*. Ewing, NJ, USA: IEEE, 2012, pp. 1–6.

[4] M. Delfino, "Distributed computing," in *Particle Physics Reference Library*, C. Fabjan and H. Schopper, Eds. Springer, Cham, 2020. [Online]. Available: https://doi.org/10.1007/978-3-030-35318-6_14

[5] Acumen Research and Consulting, "Cloud computing market worth usd 2,495.2 billion by 2032, growing at a cagr of 17.8

[6] "International journal of cloud applications and computing (ijcac)," vol. 5, p. 17, 2015, © 2015.

[7] S. Hwang, K. Lee, and Y. Chin, "Data replication in a distributed system: A performance study," in *Database and Expert Systems Applications. DEXA 1996*, ser. Lecture Notes in Computer Science, R. Wagner and H. Thoma, Eds. Springer, Berlin, Heidelberg, 1996, vol. 1134, published 26 June 2005. [Online]. Available: https://doi.org/10.1007/BFb0034724

[8] E. Leu, "Data structures and algorithms," arXiv:2307.12448 [cs.DS], 2023, u.S. Patent No. 11,429,452 was granted on August 30, 2022, for the invention. 11 pages, 2 figures. [Online]. Available: https://doi.org/10.48550/arXiv.2307.12448

[9] S. B. Balaji, M. N. Krishnan, M. Vajha, V. Ramkumar, B. Sasidharan, and P. V. Kumar, "Erasure coding for distributed storage: An overview," arXiv:1806.04437 [cs.IT], June 12 2018. [Online]. Available: https://arxiv.org/pdf/1806.04437

[10] M. Sasikumar, P. Jayarin, and F. Vinnarasi, "A hierarchical optimized resource utilization based content placement (horcp) model for cloud content delivery networks (cdns)," *Journal of Cloud Computing*, vol. 12, p. 139, 2023. [Online]. Available: https://doi.org/10.1186/s13677-023-00519-2

[11] H. Isah, T. Abughofa, S. Mahfuz, D. Ajerla, F. Zulkernine, and S. Khan, "A survey of distributed data stream processing frameworks," *IEEE Access*, vol. 7, pp. 154 300–154 316, 2019.

[12] M. Cai, M. Li, and W. Cao, "Blockchain based data distribution and traceability framework in the electric information management system," *Procedia Computer Science*, vol. 162, pp. 82–87, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050919319738

[13] A. Alwakeel, "An overview of fog computing and edge computing security and privacy issues," *Sensors*, vol. 21, no. 24, p. 8226, 2021. [Online]. Available: https://doi.org/10.3390/s21248226

[14] P. Cheng, Y. Lu, Y. Du, Z. Chen, and Y. Liu, "Optimizing data placement on hierarchical storage architecture via machine learning," in *Network and Parallel Computing. NPC 2019*, ser. Lecture Notes in Computer Science, X. Tang, Q. Chen, P. Bose, W. Zheng, and J. Gaudiot, Eds. Springer, Cham, 2019, vol. 11783. [Online]. Available: https://doi.org/10.1007/978-3-030-30709-7_23

[15] G. Ausiello, M. Protasi, A. Marchetti-Spaccamela, G. Gambosi, P. Crescenzi, and V. Kann, *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*, 1st ed. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1999.

[16] Y. LeCun, C. Cortes, and C. J. Burges, "The mnist database of handwritten digits," http://yann.lecun.com/exdb/mnist/, 2010.

[17] Google Cloud, "Google cloud documentation," https://cloud.google.com/docs, 2024, accessed: 2024-06-27.

[18] Kubernetes, "Kubernetes documentation," https://kubernetes.io/docs/home/, 2024, last modified April 20, 2024 at 8:53 PM, Accessed: 2024-06-27.

[19] Docker, "Docker documentation," https://docs.docker.com/, 2024, accessed: 2024-03-21.

[20] E. Francis, "Yaml tutorial: Everything you need to get started," https://www.cloudbees.com/blog/yaml-tutorial-everything-you-need-get-started, 2023.

[21] TensorFlow Federated, "Tensorflow federated api documentation," https://www.tensorflow.org/federated/api_docs/python/tff, 2024, accessed: 2024-04-14.

[22] "Pytorch," https://pytorch.org/, 2024, accessed: 2024-06-27.

[23] ImageNet, "Imagenet website update," https://www.image-net.org/, March 11 2021, accessed: 2024-04-17.

[24] IMDb, "Non-commercial datasets," March 18 2024. [Online]. Available: https://developer.imdb.com/non-commercial-datasets/

[25] S. C. Nayak, V. Tiwari, and B. K. Samanthula, "Review of ransomware attacks and a data recovery framework using autopsy digital forensics platform," in *2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC)*, 2023, pp. 0605–0611.