Exploring Distributed Machine Learning System on Rasberry Pi Computer Cluster

By: Isaac Torres
Polytechnic University of Puerto Rico
Advisor: Dr. Alfredo Cruz, PhD

Abstract

This project explored the use of Distributed Machine Learning (DML) as a potential tool in training times of Machine Learning (ML) models in lower-end computer clusters. To provide alternatives for students and scientists when implementing their ML environment without expensive/performant hardware. As part of this, an ML training environment was developed and deployed using container technology on a raspberry pi (RPI) computer cluster. This cluster was used to train ML classifier models over the popular CIFAR10 dataset image dataset. When producing models' different configurations were used on the cluster to vary the number of nodes and processor cores used during training to analyze the behavior of training times for models in a distributed and non-distributed environment.

Introduction

Usually, when working with any nontrivial machine learning application, a significant amount of data is required. Machine learning models are valued depending on how accurately they can complete the task, and it is generally the case that models trained with higher amounts of data tend to be more accurate. Although many factors are also involved in this, a significant amount of data is needed to be processed to pursue better and more accurate models. This results in a rise of the necessary processing power required to train models in a reasonable amount of time. There are two possible ways to approach this scaling problem. The first is to perform vertical scaling. The classic example of this is adding programmable GPUs to a host system [1, 2]. The second way this can be approached is by scaling horizontally. This is where distributed machine learning systems come in; these are systems and algorithms designed to take advantage of multiple computer nodes to process workloads faster than traditional machine learning strategies. This project had the purpose of viewing how distributed machine learning can be leveraged to allow lower-end devices to be used to complete nontrivial machine learning tasks. This was explored by developing a training environment/system to be used for training machine learning models. This system was used to perform various tests on a microcomputer cluster consisting of 4 Raspberry Pi computer nodes. These tests consisted of training machine learning models as classifiers on the popular CIFAR10 image dataset. This dataset consists of 60000 32x32 color images of one of ten possible classes.

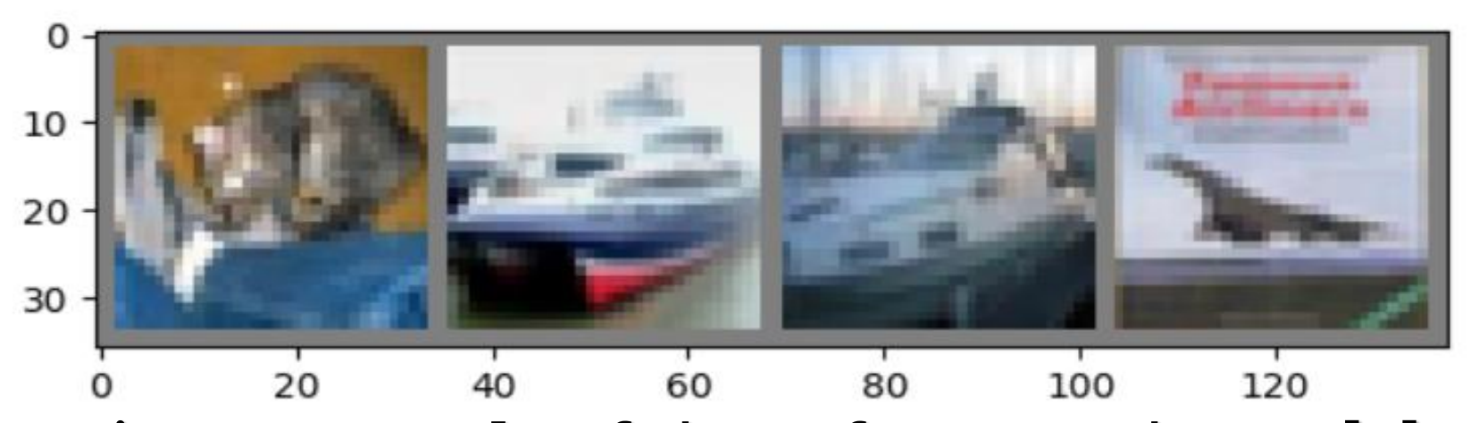

Figure 1: Sample of data of CIFAR10 dataset[3].

The system allowed different configurations to train models on the CIFAR10 test data with a varying number of nodes in the cluster and the number of processor cores used on each host processor. These values were varied to view the impact on performance. The unique combination of these served as the different tests conducted in the project.

Problem

Current literature and research suggest that there are speed benefits to training models in a distributed machine learning environment [4], thus this poses the question of whether this reduction can be leveraged to extend the possible use cases for training models in lower-end computer clusters such as those made of Rasberry Pi computers. What are the practical limitations of running distributed machine learning environment for a Raspberry Pi computer cluster? How do training times for models behave on such a cluster when adding additional nodes to the cluster. How do training times behave when increasing the amount of processing power in the cluster.

Methodology

For this project, a system or environment was required which would allow to quickly train and test machine learning models not only in a single host machine but to also be distributed through multiple hosts. The developed environment consists of a combination of tools and code. The system leveraged container technology to package the source code and dependencies into deployable packages (docker images/containers). This included the code for two ML learning algorithms, one used for traditional ML and one for DML. These executable packages could be deployed to the Raspberry Pi cluster (as shown in Figures 2 & 3) using available container orchestration tools.

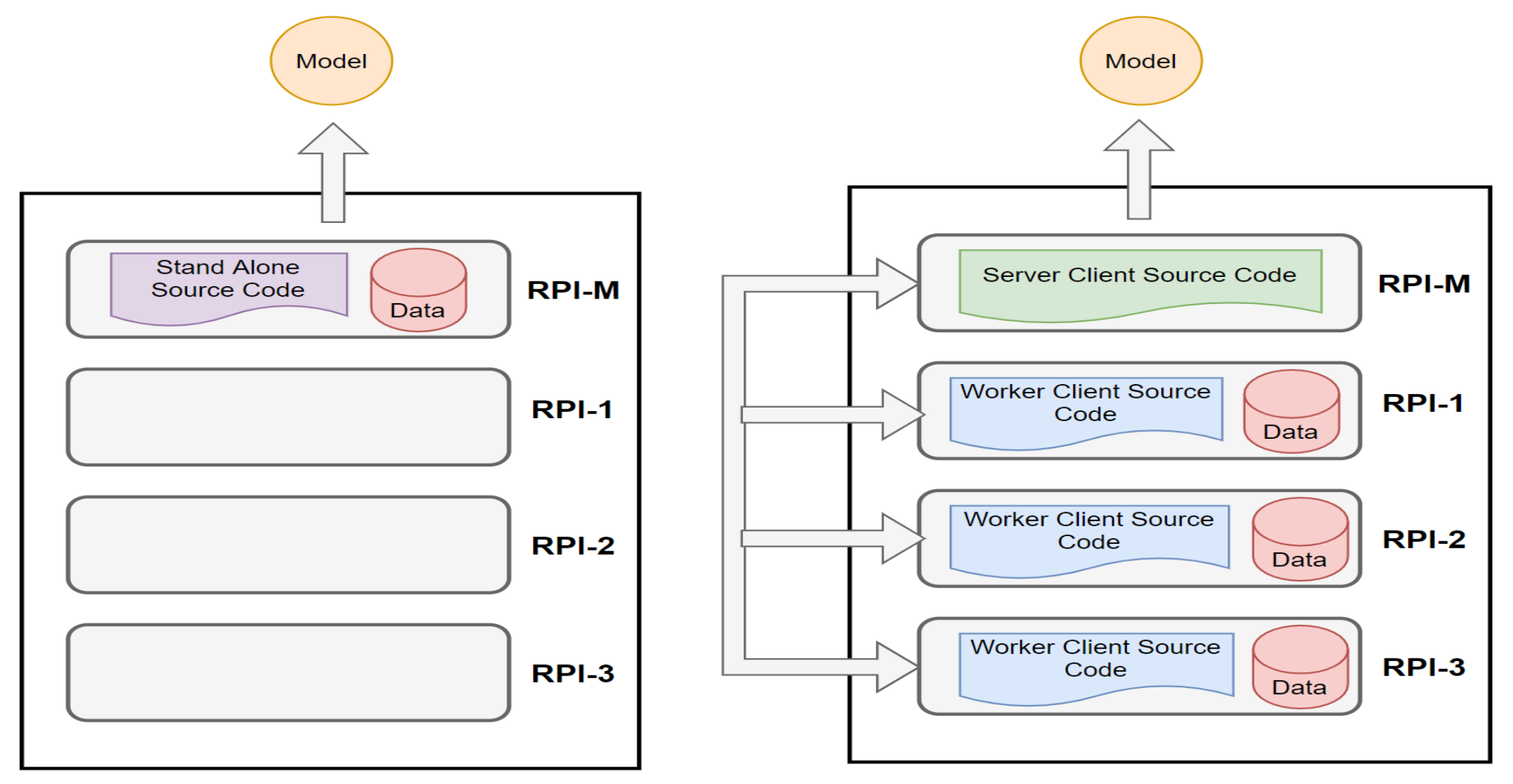
Model
Stand Alone Source Code
Data
RPI-M
RPI-1
RPI-2
RPI-3
Model
Server Client Source Code
RPI-M
Worker Client Source Code
Data
RPI-1
Worker Client Source Code
Data
RPI-2
Worker Client Source Code
Data
RPI-3

Figure 2: Visualization of source code deployed on Raspberry Pi Cluster.

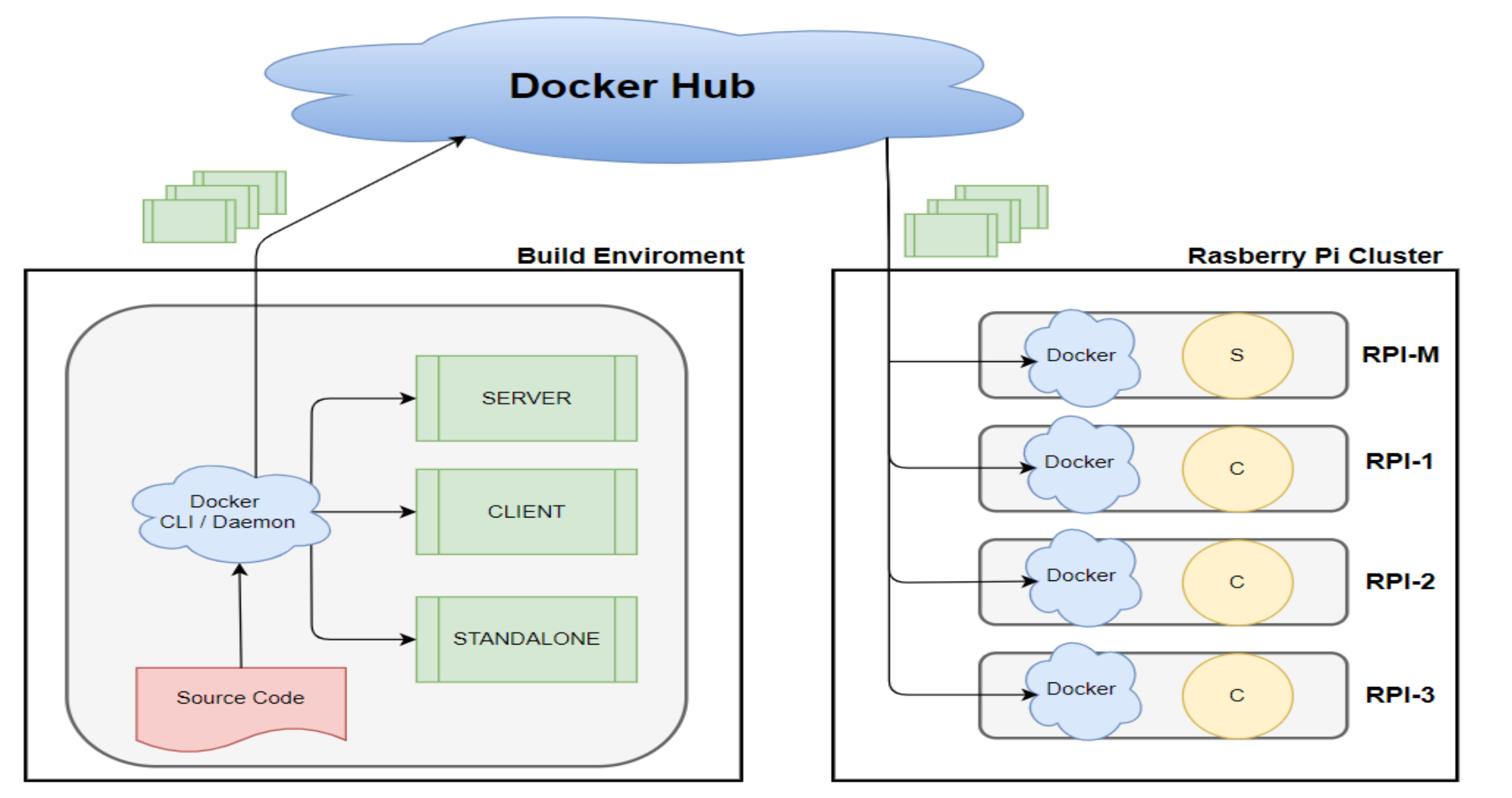

Figure 3: Visualization of packaging source code for deployment on Raspberry Pi Cluster.

The deployment process was modified when running different test cases. This was accomplished through the creation of configuration files that could be used to define various constraints for containers as they were executed. A separate configuration file was required for each test case, varying 1 to 3 Nodes and 1 to 6 of available processors. One node of the cluster was always reserved to act as the master node which handled coordination when using multiple nodes.

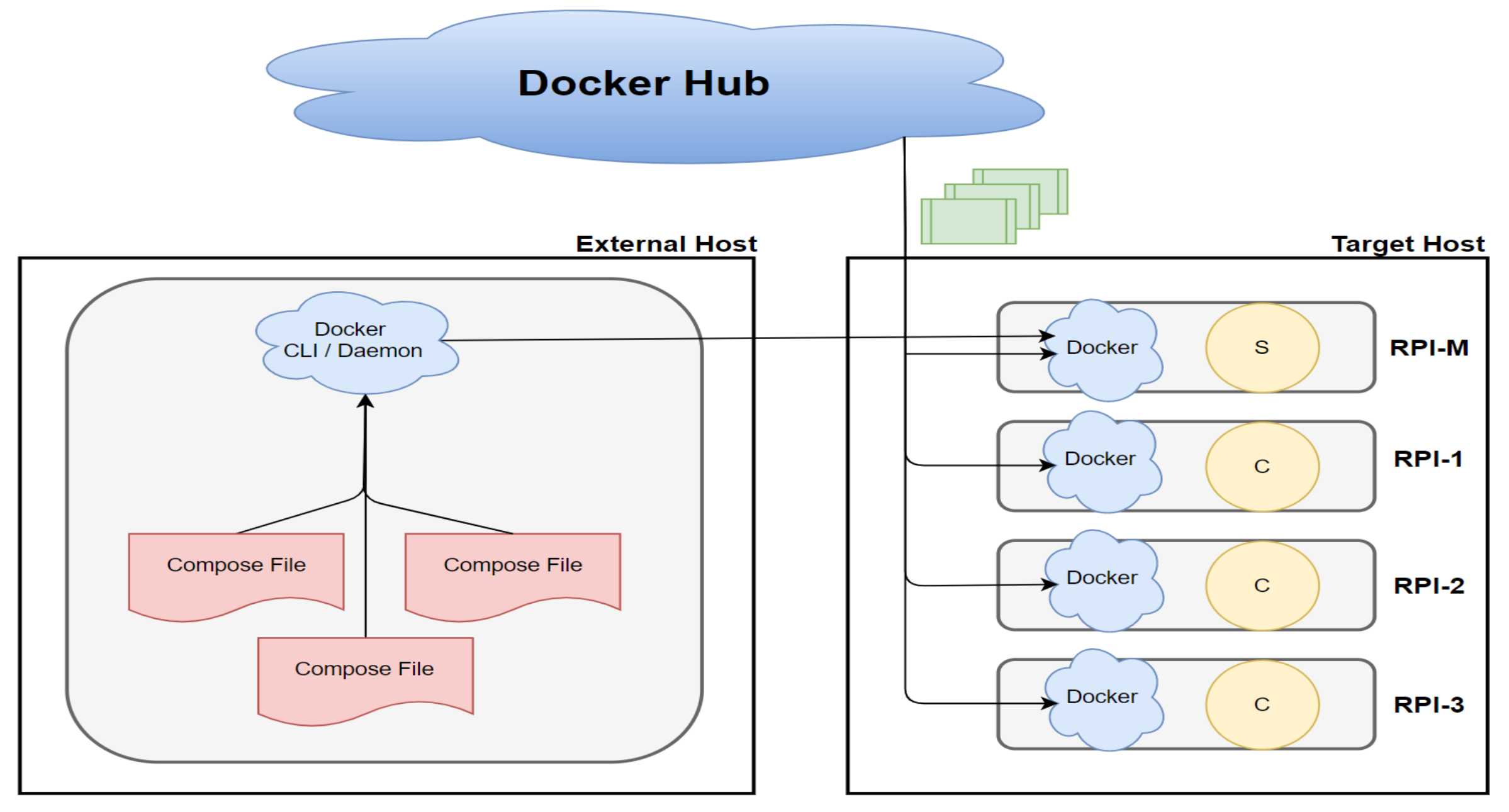

Figure 4: Visualization of modifying deployment parameters for packages through compose files.

Results & Analysis

From the results, various trends regarding the recollected data could be seen. First, training times for models in a distributed environment outpaced training times for the non-distributed tests. When using three worker nodes for example the models were trained 2.64 times faster than training in a single RPI using traditional methods.

Using additional nodes for test cases was not without consequences, when using multiple nodes in test cases the amount of overhead observed when training models increased with the number of nodes used. Extrapolating from observed data clusters with higher than 5 nodes would result in more time spent coordinating nodes than actually performing work when training a model. Reducing the potential benefit of adding additional nodes.

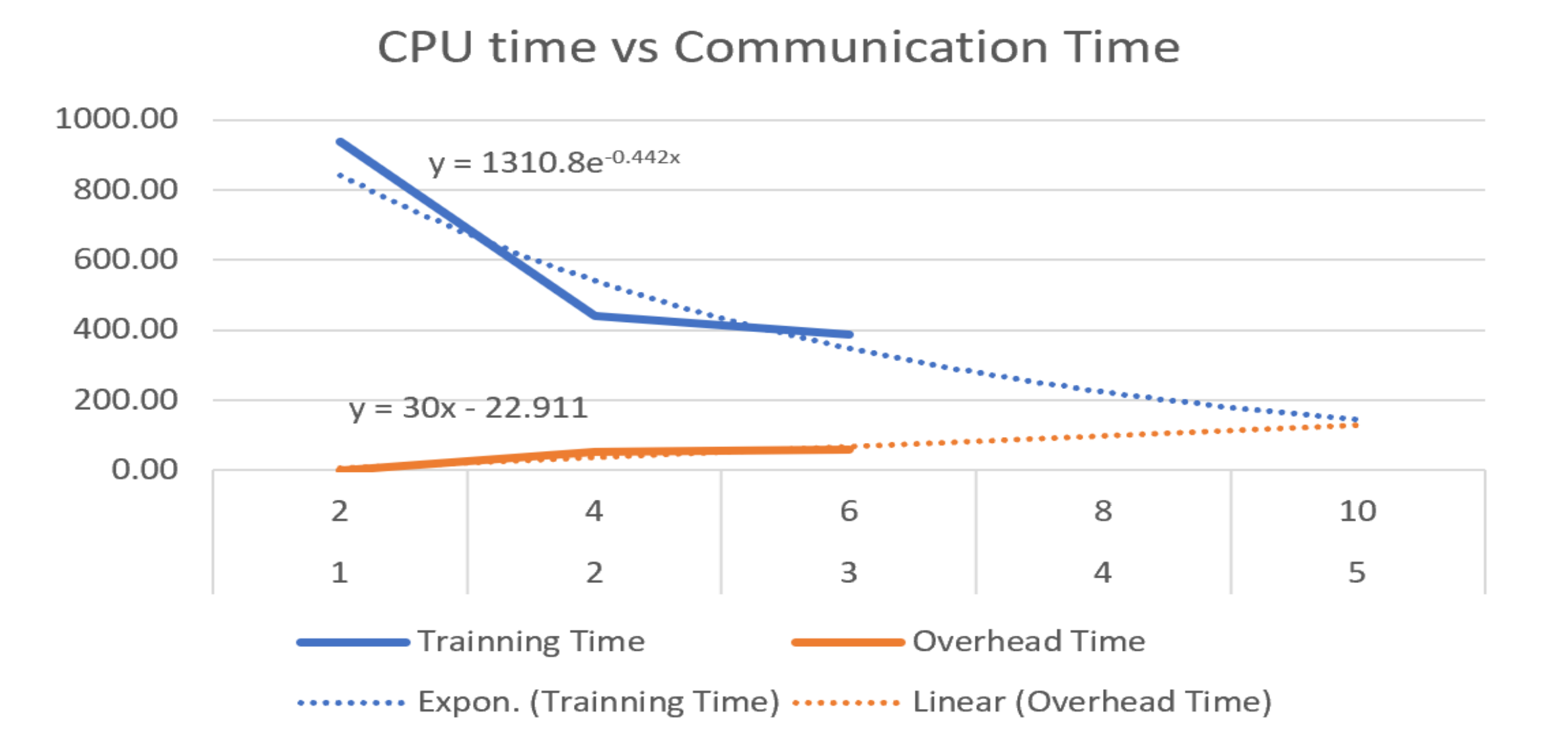

Figure 5: Training times and recorded overhead for training models varying the allowed node count in the cluster.

A similar pattern can be seen when comparing speedup values from test cases where the number of nodes varied and those where the node count remain constant, but the core count varied. In the later speedup, values showed diminishing returns meanwhile in the former speedup values for adding additional cores grew steadily as they were added.

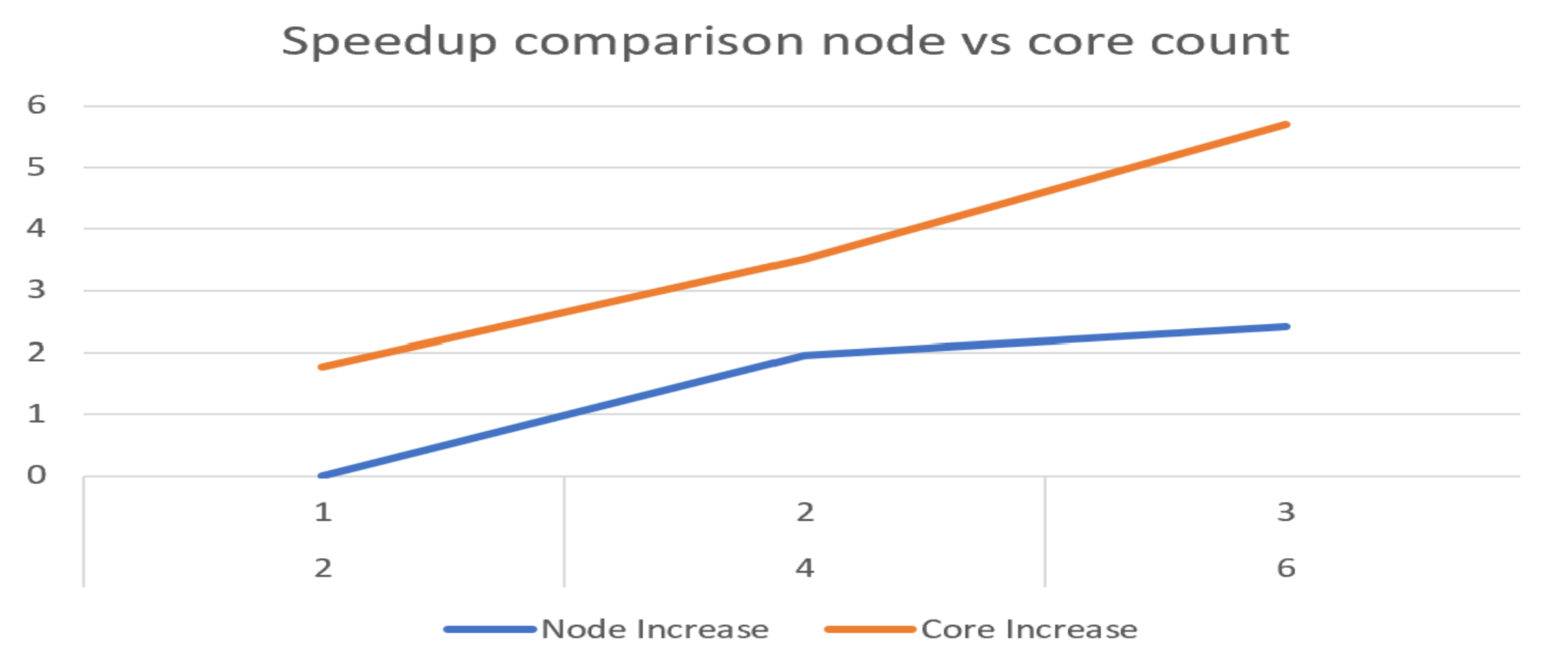

Figure 7: Visualization of recorded speedup values when varying the node count used in training models vs speedup values observed when increasing equivalent core counts in a constant three node configuration.

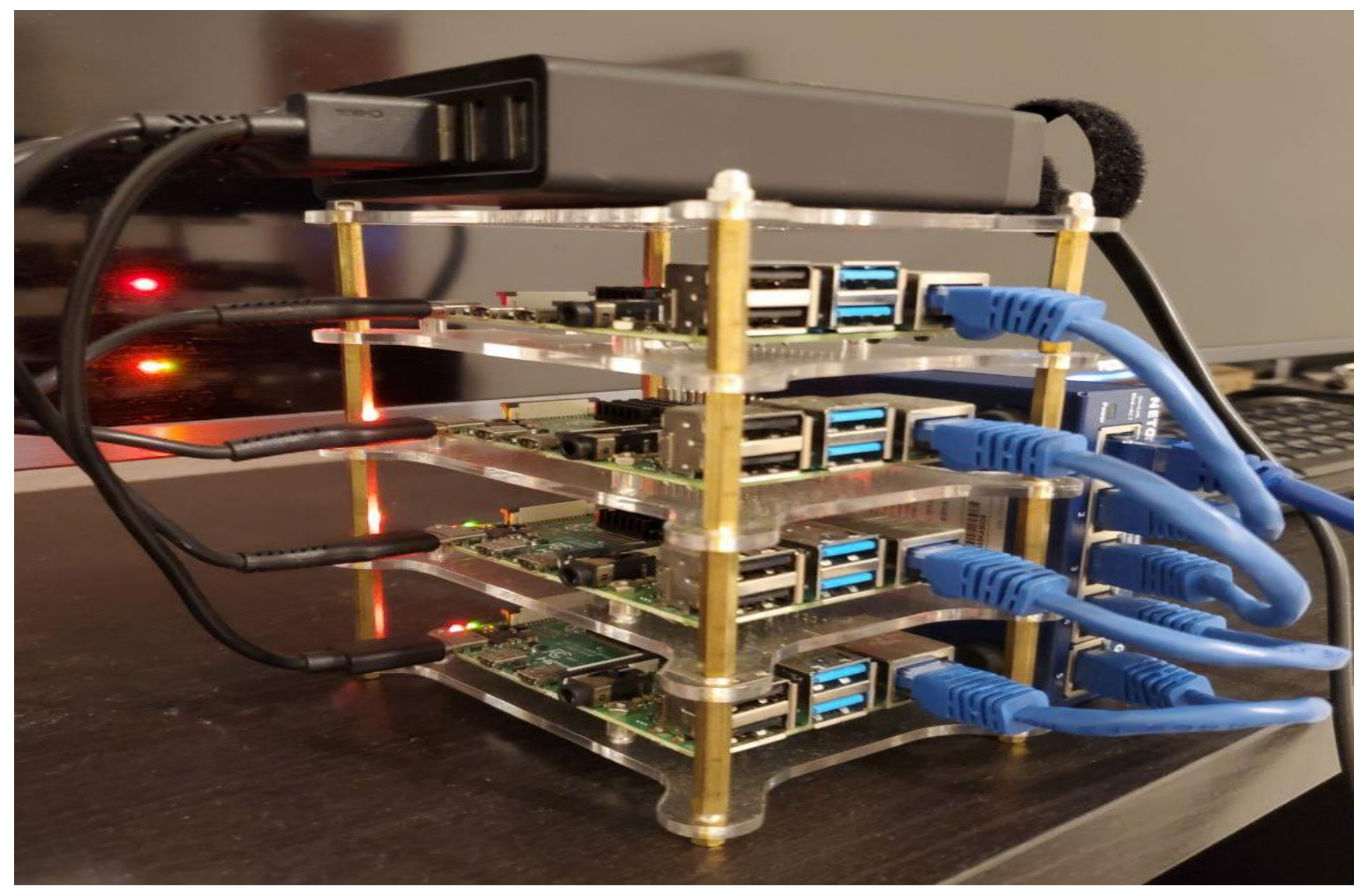

Figure 8: Image of Raspberry Pi Cluster used in project.

Conclusion

For this project, an environment for training machine learning models with the functionality to scale up or down with relative ease was successfully created. This system was used to examine the performance of an RPI cluster in training classifier models over the CIFAR10 dataset. By observing training times produced by the test cases, it was found that in general, models produced using a distributed approach were trained in less time than models trained with an undistributed approach. Additionally, when examining the effect of adding additional cores to the system without the added complexity of adding additional nodes it was found that this could result in greater speedup values when adding the equivalent processing power of an additional node to the system. It is apparent that training machine learning models was feasible in an RPI. Although examining recovered data it was observed that training time could not be reduced indefinitely by adding additional nodes to the cluster, due to diminishing returns. For this implementation, the max practical number of RPI that could be used in the cluster was found to be from 4 to 5. Since RPI are not traditionally designed to have their hardware be upgraded, this creates a hard limit for workloads able to be run on RPI clusters.

Future Work

The biggest difficulties involved with this project revolved around the hardware limitations of the individual Rasberry Pi computers. This resulted in slower development and testing times for the project. When selecting the tools and software used in this project the focus was placed on tools that were simple to implement and allowed for faster prototyping this came at the consequence of using tools that require more processing power to operate in comparison to more lightweight tools. Future work would include a revision of the code used to implement more lightweight software and reduce the number of tools used such as docker to analyze the impact on this could have on training times. Additionally, alternate datasets and types of machine learning problems should be tested to verify the results across different machine learning algorithms.

Acknowledgements

This work is supported by, or in part by, the DoD Cybersecurity Scholarship Program (CySP) under grant # H98230-20-1-0355. Special thanks are extended to Dr. Alfredo Cruz, Ph.D. of the Polytechnic University of Puerto for his extended supervision, guidance, and patience throughout this project.

References

[1] H. Kargupta, et al, “MobiMine: Monitoring the Stock Market From a PDA,” *ACM Explore. News.*, vol. 3, no. 2, pp. 37–46, 2002.

[2] H. Kargupta, et al, “VEDAS: A Mobile and Distributed Data Stream Mining System for Real-time Vehicle Monitoring,” In Proceedings of the 2004 SIAM International Conference on Data Mining, Lake Buena Vista, FL, USA, pp. 300–311, 2004.

[3] A. Krizhevsky. (2013). *CIFAR-10 and CIFAR-100 Datasets*. [online] Available: https://www.cs.toronto.edu/~kriz/cifar.html

[4] J. Verbraeken, et al. (2020). *A Survey on Distributed Machine Learning*. [online] Available: https://dl.acm.org/doi/fullHtml/10.1145/3377454