WIP: How Latency Differences between Workers Affect Parameter Server Training

Leonard Paeleke
Hasso-Plattner-Institute
Internet-Technologien und Softwarization
Potsdam, Germany
leonard.paeleke@hpi.de

Holger Karl
Hasso-Plattner-Institute
Internet-Technologien und Softwarization
Potsdam, Germany
holger.karl@hpi.de

ABSTRACT

Distributed Machine Learning (ML) methods such as the well-known Parameter Server (PS) are often used in heterogeneous environments, e.g., when performing edge computing. In heterogeneous environments, model training is affected by differences in link latency, computing resources, and data distribution. We compare synchronous and asynchronous PS training and show that low latency is tolerable for asynchronous PS.

ACM Reference Format:
Leonard Paeleke and Holger Karl. 2023. WIP: How Latency Differences between Workers Affect Parameter Server Training. In *Proceedings of Networked AI Systems (NetAISys' 1).* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Machine Learning (ML) handles increasing model size and training requirements by *distributed* ML [1], e.g., using a Parameter Server (PS) [2, 3] to distribute computation across multiple decentralized machines (*workers*) that exchange update gradients with a centralized machine (*parameter server*). The parameter server stores the model parameters, manages the data distribution, and aggregates and applies the update gradients received from workers. Each worker maintains an instance of the model and usually accesses only non-overlapping portions of the data. The complete data set thus distributes among all workers and can either be transferred from central storage, e.g., the parameter server, or collected and stored directly by the workers. Each worker processes its data to determine update gradients that describe the necessary changes to the model parameters (weights, biases) for a better prediction of its data. For computing the updates, workers typically apply stochastic gradient descent (SGD). After processing a predefined number of data samples, the workers share their update gradients with the parameter server and then receive the updated mutual model. How many other workers have contributed to the mutual model instance depends on the PS implementation. In synchronous PS implementations, for a mutual model update, the parameter server aggregates the update gradients from all workers simultaneously before broadcasting it. Usually, the workers interrupt their processing until receiving the updated model. In contrast, in asynchronous PS implementations, the parameter server updates and returns the updated model immediately after receiving an update gradient [4].

PS is exposed to varying link latency or computing capacities. Both affect the order of model updates, impacting model development. For example, edge computing often takes place in such heterogeneous environments, leading to long training times for synchronous PS, which motivated asynchronous PS [4]. However, asynchronous PS can hamper model convergence as workers process different model instances [5].

To attenuate these effects, various methods have been described, e.g., back-up workers [5], limiting worker staleness [6], or changes to SGD [7, 8]. However, no nostrum exists for handling staleness effects in training. With our study, we aim to help understand how update delays, e.g., due to heterogeneous link latency or computing capacity, affect model development and determine what delays can be tolerated. Our key finding is that these effects are very different for independent and identical distributed (IID) and non-IID distributed training data: if workers set statistically different parts of the training set, asynchronous training becomes much more susceptible to such infrastructure heterogeneity; synchronous training, on the other hand, stays robust but slow. In the following, we show our ongoing work to evaluate the impact of link delays on the evolution of ML models trained with PS. We take an experimental approach to provide meaningful results, reflecting the full complexity of actual system setups.

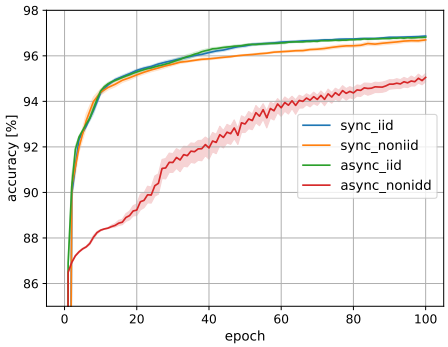
2 EXPERIMENT DESIGN

In our experiment, we train an artificial neural network on the NSL-KDD data set [9] using either synchronous or asynchronous PSs. The training infrastructure comprises two workers connected to a parameter server. We implemented the two PS methods with Ray [10], connecting three virtual machines (Ubuntu 22.04) on the same physical server with equal computing resources. To model update delays due to varying link latency or computation capacity, we impose additional link delays from worker to parameter server using the `tc` package. This allows us fine-grained control over the time from an updater request from the parameter server to a worker to the time the update does arrive at the parameter server. We call a setting *homogeneous* if all virtual machines have the same computing resources and link delays; it is *heterogeneous* if `tc` adds varying delays to the links between PS–Worker 1 and PS–Worker 2.

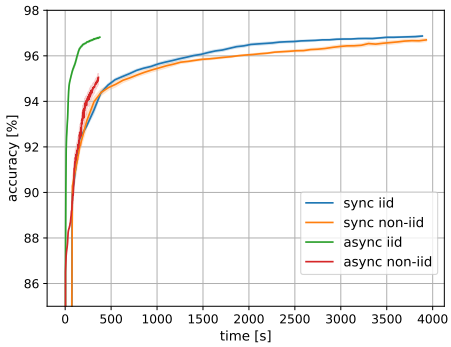
The NSL-KDD is a network traffic dataset typically used for training intrusion detection systems. This is expected to be a distributed ML application in future mobile networks [11]. Under supervised

NetAISys' 1, June 2023, Helsinki, Finland

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn



(a) Accuracy over epochs



(b) Accuracy over time

Figure 1: Comparison of synchronous and asynchronous Parameter Server with IID and non-IID data. Training for 100 epochs. 100ms delay difference between workers.

learning, models learn to classify traffic into five classes: Denial of Service (DoS), User to Root (U2R), Remote to Local (R2L), Probing, and Normal. Here, the model is trained with three-quarters of the data set, while the remainder is used to evaluate the model's accuracy. As the disproportionate impact of data distribution on model development in heterogeneous settings is our key hypothesis, we split the data differently for IID and non-IID experiments. In IID experiments, each worker accesses one-half of the training set containing all classes. In the non-IID experiments, Worker 1 accesses the DoS, R2L, and one-half of the Normal traffic data, and Worker 2 accesses U2R, Probe, and the other half of the Normal traffic data.

The model is a fully-connected artificial neural network with one hidden layer. The input layer has 93 neurons, the hidden layer 21 neurons, and the output layer five neurons, one for every class as a one-hot-encoding. The hidden-layer output is ReLu activated; the output layer uses a logarithmic softmax activation.

3 HOW LATENCIES IMPACT MODEL DEVELOPMENT IN PARAMETER SERVER

In this section, we analyze the effects of varying latency between PS and workers on model development. We consider all four combinations: model training with synchronous and asynchronous PS and IID and non-IID data. Initially, the link delays are 1 ms for PS–Worker 1 and 100 ms for PS–Worker 2. The additional delays are applied uniformly. We repeated each training twenty times with reshuffled training set and different parameter initializations of the model. Model accuracy is the key metric, and we show it over both number of training epochs as well as over wall-clock time.

Figure 1 shows that the accuracy is comparable between synchronous PS with both data distributions and asynchronous PS with IID data distribution. The advantage of asynchronous PS is evident when comparing the wall-clock time: although all three setups iterate over the data set just as often and compute the same amount, the asynchronous PS is about 10x faster than the synchronous PS. This is to be expected in such a vastly heterogeneous setup. However, for non-IID data, models trained by asynchronous PS require more iterations over the data set to achieve similar results.

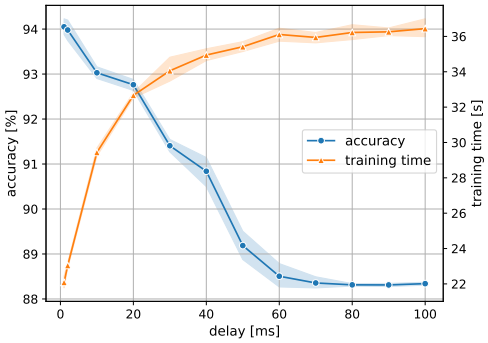


Figure 2: Accuracy and training time after 10 epochs for asynchronous Parameter Server over worker delay differences.

These results elucidate the big impact of the combination of non-IID data with asynchronous training and question the viability of this approach. To deepen the understanding, we repeated the asynchronous PS training with non-IID data for different delay differences. We keep the link delay between PS and Worker 1 fix at 1 ms, and solely vary the uniform link delay between PS and Worker 2. Figure 2 shows that lower delay differences increase model accuracy while reducing training time. For delays greater 60 ms, the effect appears to attenuate and become asymptotic.

4 SUMMARY AND OUTLOOK

We have shown that for the well-known PS method, latency affects model development in terms of both training time and accuracy. In addition, we found for asynchronous PS when data are non-IID, the accuracy decreases at high latency differences. On the other hand, for low latency, asynchronous PS is clearly superior.

We plan to extend this work in progress, short-term, to a quantitative characterization of the interaction between degrees of "non-IID-ness" (e.g., correlation between batches) arising from different

data sets and variety of latency/CPU resources among works regarding metrics like wall-clock-time or total effort. We also want to look at how susceptible different distributed ML architectures (e.g., All-Reduce or Federated Learning) are to these effects. Mid-term, we see a need to come up with a scheme to dynamically select the best training approach, e.g., switching between synchronous and asynchronous methods when system parameters like latency change.

ACKNOWLEDGMENTS

The authors acknowledge the financial support by the Federal Ministry of Education and Research of Germany in the project "Open6GHub" (grant number: 16KISK011).

REFERENCES

[1] J. Sevilla, L. Heim, A. Ho, T. Besiroglu, M. Hobbhahn, and P. Villalobos, "Compute trends across three eras of machine learning," in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022, pp. 1–8.

[2] A. Smola and S. Narayanamurthy, "An architecture for parallel topic models," *Proceedings VLDB Endowment*, vol. 3, no. 1-2, pp. 703–710, Sep. 2010.

[3] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server," 2014.

[4] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng, "Large scale distributed deep networks," *Advances in neural information processing systems*, vol. 25, 2012.

[5] J. Chen, X. Pan, R. Monga, S. Bengio, and R. Jozefowicz, "Revisiting distributed synchronous sgd," *arXiv preprint arXiv:1604.00981*, 2016.

[6] Q. Ho, J. Cipar, H. Cui, S. Lee, J. K. Kim, P. B. Gibbons, G. A. Gibson, G. Ganger, and E. P. Xing, "More effective distributed ml via a stale synchronous parallel parameter server," *Advances in neural information processing systems*, vol. 26, 2013.

[7] J. Jiang, B. Cui, C. Zhang, and L. Yu, "Heterogeneity-aware distributed parameter servers," in *Proceedings of the 2017 ACM International Conference on Management of Data*, ser. SIGMOD '17. New York, NY, USA: Association for Computing Machinery, May 2017, pp. 463–478.

[8] X. Lian, W. Zhang, C. Zhang, and J. Liu, "Asynchronous decentralized parallel stochastic gradient descent," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3043–3052.

[9] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1–6.

[10] P. Moritz, R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, M. Elibol, Z. Yang, W. Paul, M. I. Jordan, and I. Stoica, "Ray: A distributed framework for emerging AI applications," Dec. 2017.

[11] M. Usama, J. Qadir, A. Raza, H. Arif, K.-L. A. Yau, Y. Elkhatib, A. Hussain, and A. Al-Fuqaha, "Unsupervised machine learning for networking: Techniques, applications and research challenges," *IEEE Access*, vol. 7, pp. 65 579–65 615, 2019.