
POSTER: UNDERSTANDING RESOURCE REQUIREMENTS OF FL ALGORITHMS

Sepehr Delpazir¹ Ali Anwar² Nathalie Baracaldo² Khalil Al-Hussaei¹

ABSTRACT

Federated Learning (FL) is an approach to ML that allows training a model without having to gather all the data at one place. In FL, training happens at the data source (clients) and a central server (aggregator) is used to combine the trained models using a fusion algorithm to generate a global model. While doing FL to train a model, it is important to know how much system resources such as CPU and memory needs to be reserved as processing often happens on resource constrained setups. Moreover, understanding which fusion algorithms works best with which models can also help in improving the final model accuracy. To this end, in this paper, we perform an in depth analysis of several FL training processes to understand the resource usage of both the aggregator and client sides.

1 INTRODUCTION

Data privacy, security, and confidentiality are important aspects for the users of Machine Learning (ML). Federated Learning (FL) safeguards the parties' data and ensures the privacy and confidentiality by leveraging distributed systems. In FL, multiple number of clients collaboratively train a model by first doing local training, and then passing on the results to a central server called aggregator where the local models are fused into an updated global model. Unlike traditional ML, FL does not allow for the raw data to be communicated due to privacy reasons. This ensures every client's data stays private and confidential throughout the training. While using FL to train a model, it is especially important for the individual clients to be aware of the resource requirements since most of the training happens in a resource constrained environment. Different algorithms use up different amounts of system resources and being aware of the CPU and memory usage is essential for ensuring that a smooth training process takes place.

To best of our knowledge, there is no research available that explores the system resource requirements for running FL algorithms. There is a lack of empirical analysis that highlights resource requirements of a machine/device attempting to run FL. It is necessary for resource-constrained machines to have a reference of what to expect when training against

¹Department of Computer Sciences, Rochester Institute of Technology Dubai, UAE ²IBM Research, Almaden, CA, USA. Correspondence to: Sepehr Delpazir <sd3290@rit.edu>, Ali Anwar <ali.anwr2@ibm.com>.

% CPU / % Memory Usage - IoT/Edge Setting			
Fusion Algorithm	MNIST	ADULT	CIFAR-10
FedAvg	3.8 / 94	-	-
Keras Classifier	4.4 / 94	-	3.4 / 86.34
Sklearn Logistic Class.	4.7 / 86	2.5 / 86	-
Coordinate Median	6.0 / 95	-	-
Keras Gradient	9.9 / 98	-	-
Zeno	10.5 / 97	-	-
Naive Bayes	-	2.1 / 86	-

Table 1. % CPU and % memory usage while training Keras-based models for IoT/Edge setup for different data sets. We studied only those combinations that IBMFL supported.

different fusion algorithms.

In this paper, we study the system resource requirements of FL. We run multiple experiments in an effort to better understand the CPU and memory usage of training different datasets when combined with multiple fusion algorithms. The results of these experiments will hopefully give the users of FL a better idea of what to expect when training a model on their resource-constrained setups.

2 OUR APPROACH AND INSIGHTS

We perform an empirical analysis of resource usage while running IBMFL (Ludwig et al., 2020) on both IoT/Edge as well as enterprise setting. The experiments that we've carried out have been conducted on two sets of devices; **Edge setting:** Jetson AGX Xavier Developer Kit (Nvidia, 2018) powered by the NVIDIA Xavier processor. These nodes have an Aarch64 architecture, 512 core Volta GPU, an 8-core 64-bit CPU, and a 32GB memory capacity. Our setup included one aggregator node and six client side nodes. **Enterprise setting:** Lenovo ThinkPad desktop powered by the Intel Core i5 x64-based processor. This host has an 8GB RAM and is a type 64-bit operating system. This setup included one aggregator and two clients.

% CPU / Memory (MB) Usage - Enterprise Setting			
Fusion Algorithm	Keras	Pytorch	SKlearn
FedAvg	3.2 / 320	-	-
Coordinate Median	3.8 / 363	5.6 / 370	1.9 / 270
Iterative Avg.	3.5 / 372	-	-
Shuffle Iterative Avg.	3.2 / 372	-	-
PFNM	24.1 / 320	32 / 313	-
Zeno	6.5 / 444	-	-
Krum	5.5 / 350	-	-

Table 2. % CPU and absolute memory (MB) usage while training the MNIST data set for enterprise setup for different underlying ML frameworks. For Pytorch and Sklearn models, we studied only those fusion algorithms that IBMFL supported.

% CPU / % Memory Usage - IoT/Edge Setting		
Model	MNIST	ADULT
Keras	5.14 / 98	-
Sklearn	2 / 48	-
Naive Bayes	-	11.39 / 54

Table 3. Client side % CPU and % memory usage for IoT/Edge setup for different data sets.

We use 3 different datasets MNIST (Deng, 2012), Cifar-10 (Krizhevsky, 2009), and Adult dataset (Dua & Graff, 2017). For Aggregator side we use FedAvg (McMahan et al., 2017), Keras Classifier, Sklearn Logistic Classifier, Coordinate Median (Yin et al., 2018), Keras Gradient Aggregation, PFNM (Yurochkin et al., 2019) and Zeno (Xie et al., 2019) fusion algorithms.

Table 1 reports CPU and memory usage of the Aggregator side for IoT/Edge setup when training three different datasets for studied fusion algorithms. We observe that complex fusion algorithms like Zeno require almost 3x more CPU than simple fusion algorithms (e.g., FedAvg) even for a smaller dataset such as MNIST. We also observe memory usage increase for complex algorithms. Similarly, Table 2 shows CPU and memory usage for different model types for studied fusion algorithms under enterprise setup. For this experiment we use simple MNIST dataset and again observe the same trend.

When it comes to collecting CPU and memory usages on the client side, different fusion algorithms within the same FL model do not affect the system metrics differently. As a result, a singular fusion algorithm from each model (FedAvg for Keras, Sklearn Logistic Classifier for Sklearn, Naive Bayes Classifier for Naive Bayes) is chosen and the metrics are collected. As shown in Table 3, training MNIST dataset with Keras requires almost 100% more resources compared to Sklearn.

Insights: Our findings from the carried out experiments yield interesting results which include:

- There are major differences in CPU usage of fusion algorithms such as Fedavg uses half the CPU compared to PFNM on the aggregator side;

- Using different underlying ML framework/model results in a noticeable difference in memory usage. For example, Sklearn with Logistic Classification requires 10% less memory compared to Keras-based Gradient Aggregation;
- There is a huge fluctuations in CPU usage for different fusion algorithms. For example, Naive Bayes requires double the CPU usage compared to Keras with FedAvg on the client side;
- There are major differences in memory usage as well such as Keras-based Fedavg requires 100% more memory compared to Sklearn-based Logistic Classification on the client side.

REFERENCES

- Deng, L. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.
- Dua, D. and Graff, C. UCI machine learning repository, 2017. URL <http://archive.ics.uci.edu/ml>.
- Krizhevsky, A. Learning multiple layers of features from tiny images. 2009.
- Ludwig, H., Baracaldo, N., Thomas, G., Zhou, Y., Anwar, A., Rajamoni, S., Ong, Y., Radhakrishnan, J., Verma, A., Sinn, M., et al. Ibm federated learning: an enterprise framework white paper v0. 1. *arXiv preprint arXiv:2007.10987*, 2020.
- McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.
- Nvidia, N. Jetson AGX Xavier Developer Kit User Guide — Nvidia developer, Sep 2018. URL <https://developer.nvidia.com/embedded/dlc/jetson-agx-xavier-developer-kit-user-guide>.
- Xie, C., Koyejo, S., and Gupta, I. Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance. In *International Conference on Machine Learning*, pp. 6893–6901. PMLR, 2019.
- Yin, D., Chen, Y., Kannan, R., and Bartlett, P. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*, pp. 5650–5659. PMLR, 2018.
- Yurochkin, M., Agarwal, M., Ghosh, S., Greenewald, K., Hoang, N., and Khazaeni, Y. Bayesian nonparametric federated learning of neural networks. In *International Conference on Machine Learning*. PMLR, 2019.