Balancing Federated Learning Trade-Offs for Heterogeneous Environments

Matt Baughman*, Nathaniel Hudson*†, Ian Foster*†, and Kyle Chard*†
*Department of Computer Science, University of Chicago; Chicago, IL, United States
†Data Science & Learning Division, Argonne National Laboratory; Lemont, IL, United States
{mbaughman, hudsonn, foster, chard}@uchicago.edu

***Abstract*—Federated Learning (FL) is an enabling technology for supporting distributed machine learning across several devices on decentralized data. A critical challenge when FL in practice is the system resource heterogeneity of worker devices that train the ML model locally. FL workflows can be run across diverse computing devices, from sensors to High Performance Computing (HPC) clusters; however, these resource disparities may result in some devices being too burdened by the task of training and thus struggle to perform robust training when compared to more high-power devices (or clusters). Techniques can be applied to reduce the cost of training on low-power devices, such as reducing the number of epochs to perform during training. However, such techniques may also negatively harm the performance of the locally-trained model, introducing a resource-model performance trade-off. In this work, we perform robust experimentation with the aim of balancing this resource-model performance trade-off in FL. Our results provide intuition for how training hyper-parameters can be tuned to improve this trade-off in FL.**

***Index Terms*—Federated Learning, Heterogeneous Computing, Serverless Computing, Trade-offs**

I. Introduction

Computing devices, such as low-power sensors to high performance computing centers, collect/generate large amounts of data that can be used to train *Machine Learning* (ML) models. The task of training these ML models across the computing continuum is complicated by resource heterogeneity. Prior work has primarily focused on FL in local or datacenter environments only, leaving low-power, diverse devices more ambiguous. Given the lack of available bandwidth to train models using a centralized approach in real time, local training is necessary and requires techniques to allow for the vast differences in resource capabilities across different environments.

The *Federated Learning* (FL) paradigm [1], [2] is an attractive solution to this challenge of training ML models on decentralized data. FL is a distributed learning paradigm that tasks devices to perform local training on the same ML model architecture. The locally-updated model parameters from each of these devices are then sent to and aggregated on some control node which maintains an aggregated, global model. This newly-aggregated model is then shared once more to the participating devices and the loop repeats. Under FL, no raw data owned by the devices are communicated—introducing an immediate layer of data privacy. Challenges in FL include (but are not limited to): *(i)* disparity in CPU/GPU resources at training nodes, *(ii)* possibly unreliable communication channel, *(iii)* data/statistical heterogeneity at the training nodes, and *(iv)* strengthening privacy guarantees with additional techniques (e.g., homomorphic encryption) [3]. In addition to these challenges, the client-server infrastructure needed to conduct FL in conventional setups is often impractical due to requiring direct resource management.

There still exist potential trade-offs [4]–[6] due to the resources available at training nodes. For instance, training hyperparameters (e.g., number of epochs) can be tuned to reduce overall computation cost of training an ML model on a low-power device. However tuning these hyperparameters also affects how well the ML model performs under evaluation. While there are some initial works on performance of federated learning systems as systems scale [7], we aim to quantify performance of these systems for heterogeneous resources.

Our Contribution: To answer this question, we perform a robust set of experiments using FLoX [8] to evaluate the resource-model performance trade-offs implicit in FL across the computing continuum. Specifically, we *(i)* design a test case and accompanying experiments to investigate specific trade-offs in FL, *(ii)* use those results to demonstrate the existence and persistence of such trade-offs, *(iii)* illustrate how different approaches to managing trade-offs may have significant consequences on FL workflows, and *(iv)* make recommendations based on these findings to motivate future inquiry. Overall, we find the experiments in our environment confirm previous findings that it is always best to prioritize number of samples used per endpoint over training duration. Moreover, we show that aggregation should occur as frequently as possible for a well-balanced FL system. This exploration of trade-offs can be used as a starting point to further optimize FL processes among heterogeneous, pervasive computing devices.

II. Background

While traditional FL can work across many different resources, resource-level configuration makes manually scaling workloads and incorporating new resources untenable. Additionally, geographic disparities negate efforts for individually managed or highly interconnected FL environments. A candidate solution to the aforementioned infrastructure challenge is serverless computing. Since the introduction of Amazon Lambda in 2014 [10], the concept of cloud-hosted *Function-as-a-Service* (FaaS) platforms has enabled significant

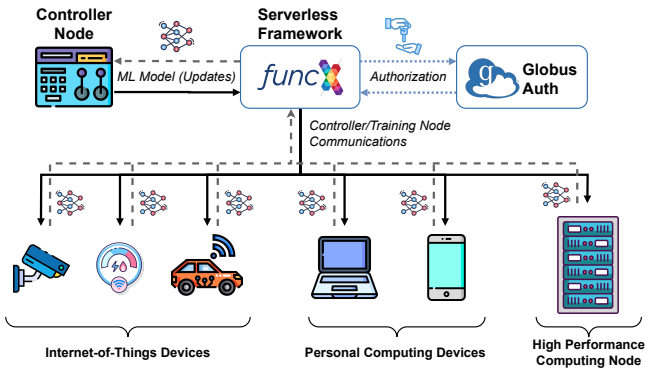

Fig. 1: Visualization of the serverless computing system we consider for design of our experiments. The system is comprised of a controller node and heterogeneous training nodes, connected through a serverless computing framework. For this, we use funcX [9].

advancements in hardware-agnostic computing. For the FL use case, this is especially relevant due to the inherit system and resource heterogeneity among training nodes.

In this work, we are motivated by the significant increase in available resource heterogeneity with the advent of serverless-based FL. To demonstrate the reasoning for the experiments and research questions presented in this paper, we present here a brief overview of the serverless FL framework we are developing, the trade-offs it presents, and the system design we developed to help gather information on these trade-offs.

A. System Description

In this work, we consider a simple serverless computing system consisting of a single controller node and a set of training nodes with locally collected and generated to train an ML model. The training nodes can be any pervasive computing device (e.g., sensors, IoT device, high performance computing nodes). The controller node is responsible for initiating the FL process on the training nodes that will perform local training. As the training nodes update their local copies of the global model (initially sent by the controller node), they will send back their updates to the controller for aggregation. For aggregation, the controller node will average the learned model weights sent back from the training nodes before updating the global model which will then be shared again to the training nodes. The communication between the controller node and training nodes is done through a serverless, federated FaaS platform known as funcX [9]. A visual depiction of the system we consider can be found in Fig. 1.

B. FLoX

We have developed a FL framework that utilizes a serverless computing architecture to enable fluid FL execution across diverse compute and data resources. Given its extensibility, focus on high performance computation, and resource-agnostic execution model, we chose the funcX [9] federated FaaS framework as the backend to FLoX (Federated Learning on funcX). In past work, we have demonstrated the effectiveness of FLoX in performing standard FL tasks, yielding throughput benefits of distributed model training, and effectively incorporating results from highly heterogeneous resources [8]. However, increasing use of heterogeneous resources requires new techniques to effectively incorporate each resource into the FL environment.

III. Methods

We perform two primary sets of experiments—the first set compares the effects of aggregation frequency on final results; the second set simulates a highly heterogeneous environment with 16 different FL training nodes and analyzes how methods used to account for differences in computational abilities affect final performance. Both of these experiment sets will demonstrate trade-offs in FL—namely, those between aggregation overhead and convergence rate and those between bottlenecking systems and managing unbalanced workloads.

For training, we give each training node a subset of the CIFAR-10 [11] dataset for all of experiments as it is a common benchmark ML task. We use a *Convolutional Neural Network* (CNN) made up of the following layers: 6 convolutional layers, a fully-connected hidden layer with 256 neurons/units, and a fully-connected output layer for classification. The common ReLU activation function is used on all layers except for the final output layer, which instead uses the softmax activation function. The models are all trained with batch size 64 and use an Adam optimizer [12].

A. Aggregation Frequency

Ideally, with respect to model capability, aggregation would be performed after every epoch. However, this is not always possible and, when it is, incurring such communications overheads may not make this the most cost-efficient (with respect to model performance and time) strategy. There are several modes by which trade-offs may be introduced into an FL system requiring less frequent aggregations.

The use of highly heterogeneous resources requires not only incorporating heterogeneous hardware into an FL environment but also accounting for geographic disparities of resources. Even with stable network connections, physical separation incurs natural latencies between endpoints. Nevertheless, highly heterogeneous resources—particularly the IoT or smart-sensor style nodes that are common data collection devices—are frequently plagued by networks lacking robustness, both in way of bandwidth and of reliability.

The overheads imposed by aggregation also require consideration of frequency, particularly when using relatively low power resources. While simple aggregation algorithms such as FedAvg [1] require relatively low overheads and scale linearly, more complex aggregation algorithms may not scale as well or may require very high computational overhead. Regardless, when 10s of ms of overhead per endpoint is spread across several hundred, then the delay becomes non-trivial.

To this end, it is important to understand the trade-offs presented by different aggregation rates. Accordingly, we have

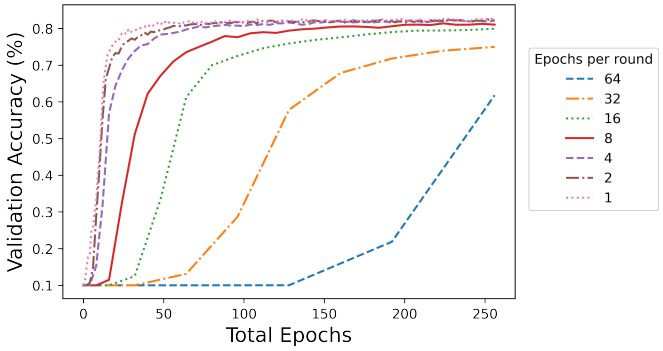

Fig. 2: Performance over time for FL workflows using different frequencies of aggregation.

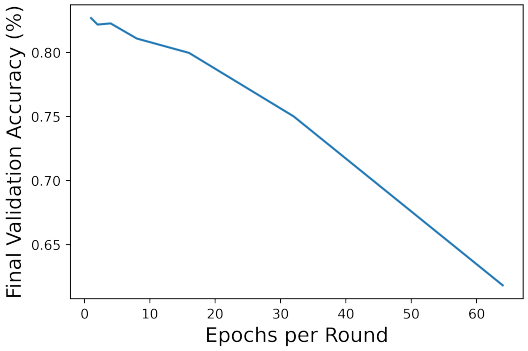

Fig. 3: Final model performance with respect to aggregation frequency.

designed and run experimental FL workflows to test to what degree these trade-offs exist and to enable initial data analyses that will allow us to navigate aggregation frequency on the fly. We build upon our prior exploratory experiments, presented in a poster [4], by using $4\times$ the number of nodes for FL, $6\times$ the number of samples per node, and using a neural network with $3\times$ the number of parameters. Effectively, this allows us to build on the previous results by *(a)* validating or challenging our existing findings regarding aggregation frequency and *(b)* extending our dataset and understanding to larger, more real-world scales with respect to the increases mentioned above.

B. FL Load Balancing

One of the primary hurdles with properly using heterogeneous resources is the necessary balancing of workloads to ensure the entire system is not bottlenecked by less-performant nodes. With the fairly linear time scaling of epochs and sample sizes in ML training, these two hyperparameters serve as two excellent variables that we can use to balance workloads. As with aggregation frequency, trade-offs are presented by reducing both samples and epochs.

Our environment, for these experiments includes 8 high-power nodes and 8 less-performant nodes—the low-power nodes are modeled as having approximately one-sixteenth the training throughput as the high-power nodes. Given this environment, we split training data evenly across all endpoints, with 3072 unique training samples available to each node.

For the experiments, we configure FL workflows to use partial epochs per round (a default of 16) and to use only some fraction of the available data. Corresponding to the relative power of the endpoints, the product of epochs per round portion and sample portion used is one-sixteenth for all experiments (e.g., 8 epochs per round is half of 16 and 768 samples is one-fourth of the available 3072, so this specific FL task will take one-sixteenth the time of the baseline). The balance configurations used are found in Table I and demonstrate different epoch-to-sample ratios as described above.

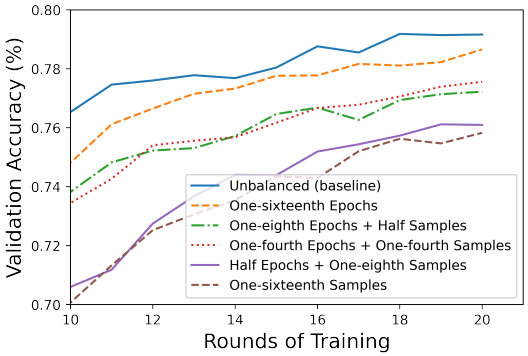

Fig. 4: Demonstration of performance in a highly heterogeneous environment with different methods of workload balancing to reduce bottlenecks.

IV. Results

We present our initial analysis of our experiments and put forward initial observations based on what we learned.

A. Aggregation Frequency

We ran 7 experiments with different rates of weight aggregation. Each configuration was run 3 times to ensure robustness against potential outliers in our results and every experiment trained for a total of 256 epochs, with aggregation frequency ranging from one epoch per round to 64 epochs per round.

From our seven experiments of different aggregation frequencies, we see the clear and expected trend that more frequent aggregation yields better results for a given level of cumulative training. In fact, we can see in Fig. 3 a near linear relationship between aggregation frequency and final accuracy. Additionally, we can observe that there is a strong correlation between rounds and accuracy more so than between total epochs and accuracy (see Fig. 2).

Finally, we also see a relatively correlative relationship between the number of epochs per round and the point of parity between the different configurations (e.g., 2 epochs per round catches up with 1 epoch per round at roughly 64 epochs, 4 epochs per round catches up at 128 epochs, and 8 epochs per round catches up at 256). The implications of this are two-fold:

TABLE I: Summary of results for different balance methods.

Balance Method	Validation Accuracy
Baseline (unbalanced)	0.792
$\frac{1}{16}$ Epochs	0.787
$\frac{1}{8}$ Epochs + $\frac{1}{2}$ Samples	0.772
$\frac{1}{4}$ Epochs + $\frac{1}{4}$ Samples	0.776
$\frac{1}{2}$ Epochs + $\frac{1}{8}$ Samples	0.761
$\frac{1}{16}$ Samples	0.758

(i) we can roughly predict how many rounds will be required to yield commensurate performance for different aggregation frequencies and *(ii)* it seems there is not necessarily a cut of in terms of low aggregation frequency where we would end up not seeing convergence. These two implications are, of course, highly speculative but definitively motivate future work on the asymptotic scaling behaviors of such FL environments.

B. FL Load Balancing

From our experiments with different ratios of epochs to samples, we found a very similar trend to our previous work. In all, final accuracy appears most dependent on the quantity of data used (see Fig. 4). In other words, our short recommendations would be to balance workloads for heterogeneous resources on epochs rather than by varying the number of samples used for training.

Additionally, we observe in our unbalanced baseline (i.e., where we train for all samples and all epochs each round on every endpoint) that our final accuracy is only 0.005 higher than the case where we balance on epochs only (see Tab. I). As the only difference between the cases here is training for many more epochs per round on each of the less-performant nodes, the combination of this results as well as our overall observation of balancing on epochs only seems to demonstrate once again the proportional lack of importance for significant training between aggregations. This also corroborates our observations in the previous section.

V. Conclusion & Future Work

This work examines the role of trade-offs in FL, the need for which is particularly amplified by shifts in FL applications and research. Specifically, we wanted to look at the different trade-offs presented by model aggregation and node-level workload balancing. To do this, we designed and ran several experiments with different FL configurations to test the potential of frequent and infrequent aggregation strategies. We also investigate the role of balancing the number of samples or epochs in order to improve system bottlenecks.

Results from these experiments demonstrated that the trade-offs we first posited in earlier works are still present in a larger FL ecosystem and, moreover, that the general trends observed in our original work hold in such an environment. Specifically, we showed that it is generally more advantageous to balance on the number of epochs rather than samples for less performant endpoints. Additionally, the more robust environment used for this experiment set allowed us to elucidate several new observations regarding these trade-offs. First, we observe that aggregation frequency is more important in a more highly distributed environment but also that rounds to convergence is relatively even across all experimental runs. Next, while we reaffirmed the importance of utilizing all available data, we also found that balancing on epochs allows us to use 53% of the compute resources to achieve $> 99\%$ of the accuracy of running all epochs and all samples unbalanced. Taking these two observations in conjunction leads us to question the necessity for continual training in between aggregation and suggests that the aggregation frequency trade-off lends itself to being examined with a more simple solution—aggregating as frequently as possible.

In future works, we intend to pursue this line of inquiry as well as investigating several other questions. We are interested in seeing if there is substantive merit to our observation regarding aggregation frequency and proportional time to convergence. We also intend to begin using this data to craft automation methods for FL to autobalance and autoconfigure workflows, enabling more efficient and less hands-on science.

Acknowledgments

This research was supported in part by DOE contract DE-AC02-06CH11357 and by NSF grants 1816611, 2004894, and 1550588.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, PMLR, 2017.

[2] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

[3] M. Aledhari, R. Razzak, R. M. Parizi, and F. Saeed, "Federated learning: A survey on enabling technologies, protocols, and applications," *IEEE Access*, vol. 8, pp. 140699–140725, 2020.

[4] M. Baughman, I. Foster, and K. Chard, "Exploring tradeoffs in federated learning on serverless computing architectures," *IEEE eScience, posters*, 2022.

[5] N. Hudson, P. Oza, H. Khamfroush, and T. Chantem, "Smart edge-enabled traffic light control: Improving reward-communication trade-offs with federated reinforcement learning," in *2022 IEEE International Conference on Smart Computing (SMARTCOMP)*, IEEE, 2022.

[6] A. Imteaj, K. Mamun Ahmed, U. Thakker, S. Wang, J. Li, and M. H. Amini, "Federated learning for resource-constrained iot devices: Panoramas and state of the art," *Federated and Transfer Learning*, 2022.

[7] J. S.-P. Díaz and Á. L. García, "Study of the performance and scalability of federated learning for medical imaging with intermittent clients," *Neurocomputing*, vol. 518, pp. 142–154, 2023.

[8] N. Kotsehub, M. Baughman, R. Chard, N. Hudson, P. Patros, O. Rana, I. Foster, and K. Chard, "FLoX: Federated learning with FaaS at the edge," *IEEE eScience*, 2022.

[9] R. Chard, Y. Babuji, Z. Li, T. Skluzacek, A. Woodard, B. Blaiszik, I. Foster, and K. Chard, "funcX: A federated function serving fabric for science," in *High-Performance Parallel and Distributed Computing*, 2020.

[10] "Amazon Lambda." aws.amazon.com/lambda/.

[11] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[12] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.