Helios: Heterogeneity-Aware Federated Learning with Dynamically Balanced Collaboration

Zirui Xu*, Fuxun Yu*, Jinjun Xiong† and Xiang Chen*
*George Mason University, †IBM Thomas J. Watson Research Center
Email: *{zxu21, fyu2, xchen26}@gmu.edu, †jinjun@us.ibm.com

2021 58th ACM/IEEE Design Automation Conference (DAC) | 978-1-6654-3274-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/DAC18074.2021.9586241

***Abstract*—As Federated Learning (FL) has been widely used for collaborative training, a considerable computational *straggler* issue emerged: when FL deploys identical neural network models to heterogeneous devices, the ones with weak computational capacities, referred to as stragglers, may significantly delay the synchronous parameter aggregation. Although discarding stragglers from the collaboration can relieve this issue to a certain extent, stragglers may keep unique and critical information learned from the non-identical dataset, and directly discarding will harm the overall collaboration performance. Therefore, in this paper, we propose *Helios* — a heterogeneity-aware FL framework to tackle the straggler issue. *Helios* identifies individual devices' heterogeneous training capability, and therefore the expected neural network model training volumes regarding the collaborative training pace. For straggling devices, a "soft-training" method is proposed to dynamically compress the original identical training model into the expected volume through a rotated neuron training approach. With extensive algorithm analysis and optimization schemes, stragglers can be accelerated while retaining the convergence for local training as well as federated collaboration. Experiments show that *Helios* can provide up to 2.5× training acceleration and maximum 4.64% convergence accuracy improvement in various collaboration settings.**

I. Introduction

In the past few years, a lot of attention is paid to "training-on-edge", which is highly promoted by intelligent edge applications. As one of the most well-recognized edge training techniques, Federated Learning (FL) expects to unit multiple resource-constrained edge devices to collaboratively train identical neural network models with their local dataset [1]. By aggregating the parameter updates from each device, a global model can be collaboratively trained efficiently and securely.

When applying FL into practical edge training, a serious problem emerges: although edge devices train the same model structure and identical workload, they may have distinct computation resources (*e.g.*, memory size, CPU/GPU bandwidth, *etc*). Therefore, like the "shortest board in barrel", the devices with extremely weak computation capacities will take much longer time for local training. These devices are referred to as *stragglers* in FL. One example is demonstrated by Fig. 1: with synchronous FL aggregation setting, stronger collaboration nodes (*Jetson Nano* and *Rasberry Pi*) have to keep an idle status to wait for the straggler (*DeepLens*) in every cycle and prolong the training cycle from 2.3 to 7.7 hours.

Intuitively, changing the synchronous FL aggregation into an asynchronous manner can directly kick out the stragglers from most aggregation cycles and accelerate the training cycles [2]–[6]. However, in FL settings where most local dataset are non-identical distributed (Non-IID), the stragglers may learn unique and critical information. And dramatically asynchronous collaboration may defect both local and global convergence performance [7], [8].

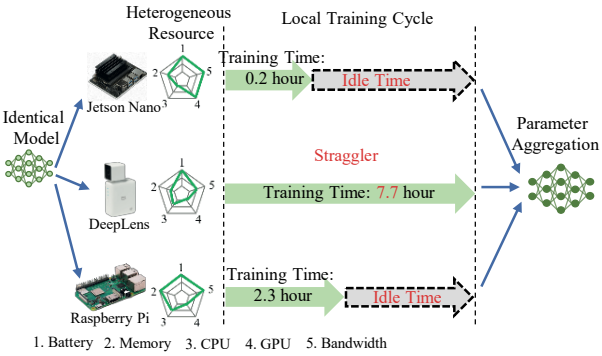

Fig. 1: The Straggler Issue in Original FL

To solve the FL straggler issue, we propose *Helios*, a heterogeneity-aware FL framework. *Helios* identifies the stragglers regarding the collaborative training pace and specify the expected neural network model training volumes. For straggling devices, a soft-training method is proposed to dynamically compress the original identical training model into the expected volume through a rotated neuron training approach. With extensive algorithm analysis and optimization schemes, the stragglers can be accelerated while retaining the convergence for local training and federated collaboration.

Our paper has the following three main contributions:

- The straggler identification methodology we proposed incorporates different approaches (*i.e.*, time-based approximation and resource-based profiling) for various FL deployment contexts, offering outstanding feasibility for FL straggler research works;
- A collaborative training method — "soft-training" is proposed to keep different part of model parameters on the stragglers alternately join the training cycles, which not only maintains synchronized FL with accelerated stragglers to preserve learning information integrity from every node, but also preserves straggler model integrity without unrecoverable compression;
- Dedicated optimization schemes are proposed based on extensive algorithm analysis and proof to enhance the overall performance.

Experiments show that *Helios* can provide up to 2.5× training speed-up and maximum 4.64% convergence accuracy improvement in various FL settings with stragglers.

This work was supported in part by NSF CNS-2003211.

Three example settings with 2 edge devices. Each device train a *AlexNet* on *CIFAR-10*

	Setting 1 (Syn.)		Setting 2 (Asyn.)		Setting 3 (Asyn.)	
Device	1A	1B	2A	2B	3A	3B
Aggregation Period.	1 Epoch	1 Epoch	1 Epoch	2 Epochs	1 Epoch	3 Epochs
Converge Accuracy	89.01%		88.89%		87.87%	

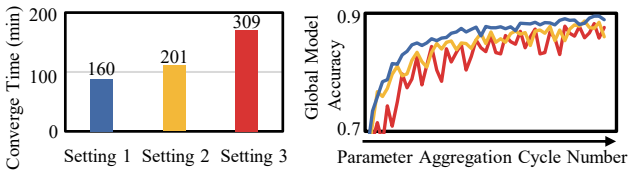

Fig. 2: Asynchronous FL Performance Evaluation

II. Preliminary

A. Federated Learning on Heterogeneous Edge Devices

In practical FL implementation, multiple edge devices collaboratively train the identical models on their local training data with two kinds of heterogeneities:

Hardware Resource Heterogeneity: Since various edge devices participate in FL, they have heterogeneous hardware resources in terms of memory size, CPU bandwidth, and *etc*. Given a certain edge device in FL, the training consumption of its CNN model can be generally evaluated as the training cycle time cost. However, the mismatches between the model's computation consumption and device's resource capacities will increase the model training time.

Information Heterogeneity: In FL process, every edge device will constantly learn the information from local training data and contribute to the final global model [9], [10]. Although lagging the FL efficiency, stragglers still have important learned information. This phenomenon becomes more prominent in the Non-IID setting [1]. The critical information learned from stragglers are distinct and cannot be ignored from the perspective of the entire collaboration process.

Based on the above two heterogeneity analysis, it is essential to eliminate stragglers' delay while preserving their learned information in FL process.

B. Federated Learning Straggler Issue

Some works are proposed to solve the straggler issue:

Asynchronous Methods: these works only aggregate parameters of stragglers at certain cycles [3], [11]–[13]. Nishio *et al.* proposed an optimized FL protocol (*i.e.*, FedeCS) to kick out straggled devices from the learning collaboration [11]. Wang *et al.* introduced a dedicated collaboration method to reduce the training loss introduced by the asynchronous stragglers [13]. Although showing acceleration performance, they cannot fundamentally eliminate stragglers and even cause information degradation and stale parameter updating [7], [8]. Fig. 2 shows an analysis for two collaborative devices under three settings. We can easily find that, the original synchronous FL achieves the best convergence accuracy. However, when the asynchronous straggler parameter aggregation cycle increases from 2 epochs (setting 2) to 3 epochs (setting 3), both the converge accuracy and speed will decrease.

Synchronous Methods: these works leveraged local training optimization methods to accelerate stragglers' training cycles so as to achieve synchronous aggregation [14], [15]. Jiang *et al.* used model pruning to compress the original models to smaller ones which can satisfy the given constraints [14].

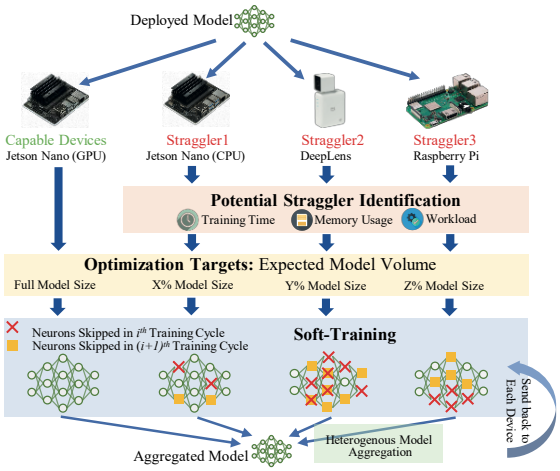

Fig. 3: The *Helios* Framework Overview

Jeong *et al.* reduced the model size in edge nodes based on knowledge distillation [15]. However, due to the permanent model structure losing, the information capacity of stragglers will decrease and may face converge degradation. As shown in [16], when introducing models with diverged structures, the global model accuracy would drop as much as 10%.

III. Framework Overview

The overall concept of *Helios* is shown in Fig. 3. In order to accelerate stragglers, the proposed framework is aiming to optimize the model training on stragglers.

Initialized Stragglers Identification: Since the straggler issue is caused by devices' heterogeneous resources, we proposed two straggler identification approaches for various FL deployment contexts to improve framework flexibility, which will be discussed in Section IV. B.

Optimization Target Determination: In Section IV. C, the initialized stragglers' optimization targets (*i.e.* the expected model volumes) are further determined, which can reduce models' training consumption and achieve acceleration.

Soft-training: Guided by the expected model structure volumes, "soft-training" method is introduced. In each training cycle, the proposed method can let different part of model parameters (*e.g.* neurons) alternately join the training, which maintains a complete model parameter updating for collaboration. As Fig. 3 shows: in the i^{th} training cycle, neurons indicated by red crosses are skipped to achieve training acceleration. However, in the $(i+1)^{th}$ training cycle, a different set of neurons (yellow squares) are skipped. Such a training method can guarantee each model parameter on stragglers has opportunities to make a contribution to the collaboration.

Optimizations for Soft-training: We further propose several dedicate optimization schemes in Section VI, which can enhance the overall convergence performance and collaboration scalability based on the extensive algorithm analysis and proof.

IV. Potential Straggler Identification in Heterogeneous Collaboration

In this section, we will discuss potential straggler identification and the corresponding optimization target determination in the heterogeneous collaboration setting.

A. *Heterogeneous Federated Learning Setup*

In the heterogeneous FL, multiple edge devices with distinct computation resources collaboratively train the identical models with their local datasets. After each local training, they upload the local model parameters to the global device for global model updating. Once obtaining a new global model, the global device sends the model parameters back to each local device. During this process, the devices with extremely weak computation capacities will take a much longer local training time, becoming stragglers in FL.

B. *Straggler Identification*

Since the straggler has a much longer training time cost than the normal devices (35× in Fig. 1), identifying it only after FL process will introduce a high time overhead. Therefore, the potential stragglers need to be initially detected by comparing the computation resources of each device. Two approaches are discussed in this paper: time-based approximation and resource-based profiling.

Time-based Approximation: If partial/all devices' hardware configurations cannot be obtained (***Black Box***), we can leverage the proposed time-based approximation to fast identify the stragglers. In this case, our approach will assign each device with a lightweight test bench (only train few iterations) and quickly record their training time cost. Then the approach ranks all devices based on their training time cost values and obtain an approximate index $T = \{T_1, T_2, ..., T_N\}$, where T_1 represents the longest time cost. Based on FL requirement, the top-k devices in the index are identified as potential stragglers.

Resource-based Profiling: If all devices' hardware configurations can be obtained (***White Box***), we can leverage the proposed resource-base profiling to accurately identify the stragglers. First, the scheme will investigate the computation resource of each device in terms of computation bandwidth, memory size, communication bandwidth, *etc.* Next, we can leverage resource models in [17] to fully profile each device's model training consumption. For example, straggler's training time cost Te can be exactly formulated by the training computation workload W, memory usage M as: $Te = W/C_{cpu} + M/V_{mc} + M/B_n$, where C_{cpu}, V_{mc}, and B_n represent device computation bandwidth, data transmitting speed, and communication bandwidth. Finally, we can identify the stragglers based on each device's computation ability.

C. *Optimization Target Determination*

After finding the potential stragglers, it is important to identify the optimization target, namely, the expected model volumes of stragglers in each training cycle to achieve acceleration. The expected model volume can be selected with a pre-define volume or adapted to a specific value according to the resource constraints.

For the first one, we can define multiple model volume levels in advanced and assign each straggler with a model volume level according to the training time cost index T. The model volume will be dynamically adjusted to an optimal point during the first several training cycles in FL process.

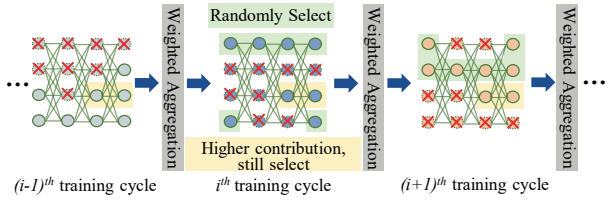

Fig. 4: Soft-Training Scheme Flow

If the model volumes of stragglers need to be accurately determined before the training, we can leverage the profiling models we built in the *Resource-based profiling*. Specifically, we select each layer with $P_i n_i$ neurons simultaneously until the model consumption approaches to the resource constraints, where n_i is the total neuron number in the i^{th} layer and P_i is a ratio between $(0, 1)$. Hence, the optimized model volumes can ensure training acceleration during FL process.

V. Soft-training on Straggling Edge Devices

With identified potential straggler and corresponding model volumes from initialization, we propose "soft-training" scheme to adapt the optimized model to specific resource constraints while guaranteeing converge effectiveness.

A. *Soft-training Process under Optimization Targets*

Fig. 4 shows the entire "soft-training" flow and we take neurons as our minimum model parameter structure. The entire "soft-training" can be divided into three steps:

Step 1. Straggler Model Shrinking: In order to meet the optimization targets (the expected model volume), we only select a set of neurons from stragglers' models to join the training cycle. As aforementioned, methods in Section IV only approximately identify stragglers and their expected mode volume. Therefore, *Helios* needs first few training cycles to finalize the stragglers and model volumes by dynamically adjusting the model volumes to accommodate normal devices *w.r.t.* training time cost.

During the partial training, two types of neurons are selected to achieve an optimized convergence performance: the neurons with the higher contributions to the collaboration convergence (primary converge guarantee) and some other random neurons (further converge optimization). Then we define the contribution metric by leverage the assumption in [18] that the neurons with larger weight parameter changing values will provide larger impacts to the global model. We assume the weight parameters of the j^{th} neuron in the i^{th} layer after training epoch S_k as $\theta^{ij}_{s,r,n}(S_k)$, and the neuron's collaboration contribution $U^{ij}(S_k)$ will be calculated by the summation of useful changes in the current training cycle as:

$$U^{ij}(S_k) = \theta^{ij}_{s,r,n}(S_k) - \theta^{ij}_{s,r,n}(S_{(k-1)}), \quad (1)$$

where a larger U^{ij} represents the target neuron has a higher collaboration contribution. We will elaborately show how to select these two types of neurons in each training cycle in Step 2 and Step 3.

Step 2. Neuron Rotation for Model Integrity: With partial training in Step 1, model training consumption on stragglers can be adapted to the specific device resource constraints.

Some previous methods leverage fixed model pruning to achieve partial training [14], [15]. However, without an optimized training algorithm to keep a rather complete model structure, the pruned model parameters will permanently lose their contribution to the collaboration, thereby inevitably hurt convergence accuracy and speed.

Therefore, in our "soft-training", we solve the conflict between the optimization targets and model integrity by letting different part of model parameters (*i.e.* neurons) alternately join training cycles. As Fig. 4 illustrates, in the i^{th} training cycle, we first choose P_s percentage neurons[1] which have the highest changing values U^{ij} and randomly select other $(1-P_s)P_in_i$ neurons from the rest $(n_i - P_sP_in_i)$ ones which have relative lower contributions. Value of P_s will be discussed in Section VI. A Therefore, the total parameter changing on a straggler is:

$$\Theta(S_k)\text{–}\Theta(S_{(k\ \ 1)})=TopK(U^{ij}) \cup Rand(U^{ij}), where\ K=P_sP_in_i, \quad (2)$$

where $\Theta(S_k)$ represents the model parameter of the straggler and $Rand$ operation is randomly selecting $(1-P_s)P_in_i$ neurons' U^{ij}. Next, in the $(i+1)^{th}$ training cycle, since the neurons with the highest contributions don't change, we still select them in the training to provide the primary converge guarantee. On the contrary, a different set of neurons with lower contributions will be selected. By alternately joining each training cycle, every neuron on a straggler has opportunity to keep model integrity.

Step 3. Aggregation and Local Model Updating: After finishing a local training cycle, each local edge device will upload the gradients of the selected neurons to the global device during the aggregation cycle. Meanwhile, the global device will average the uploaded model parameters to update the global model, which will be further discussed in Section VI. B. Once the new global model parameters are obtained, they will be sent to each local device for local model updating and the local edge devices can start the next training cycle.

With the above three steps, models on the stragglers adapted to the optimization targets while still maintain a rather complete model structure and balanced contribution, thereby guarantees the global model convergence performance.

B. Convergence Proof and Conditions Analysis

Consider federated learning with N nodes, the training dataset and loss function for each device is $\{x_n\}$ and $\{f_n\}$, respectively. Let $\Theta \in \mathbb{R}^m$ represents the model parameter and m indicates the total neuron number.

The FL optimization goal can be formulated as:

$$\min_{\Theta} \ \ f(\Theta):=\frac{1}{N}\sum_{n=1}^{N} f_n(\Theta), \ \ \Theta_{t+1}=\Theta_t-\eta_t g_t(\Theta_t), \quad (3)$$

where t is the current training epoch number (equals to S_k in Eq. 2), and $g_t(\cdot)$ represents the gradients.

Proposition-1: *Convergence Bounded by Gradient Variance*. Based on previous work [19], the global model convergence loss is proved to be bounded by the variance of gradient $g_t(\Theta_t)$ across different nodes:

[1]We empirically evaluate overhead of such sorting operation and find it can be ignored compared to the training cost (*e.g.* 18ms vs 12mins).

$$\mathbb{E}\left[f(\Theta_{t+1})\right] \le f(\Theta_t) - \eta_t \left\|\nabla f(\Theta_t)\right\|^2 + \frac{L}{2}\eta_t^2 \mathbb{E}\left\|g_t(\Theta_t)\right\|^2. \quad (4)$$

Therefore, in our soft-training algorithm, we regulate the third term, which is the gradient variance to achieve the desired model convergence. Such theoretical convergence boundary can be also applied to Non-IID situation, where the gradient variance $\mathbb{E}\left\|g_t(\Theta_t)\right\|^2$ can still find a upper boundary.

Proposition-2: *Bounded Gradient Variance in Soft-Training*. The gradient variance in soft-training is bounded by $(1+\rho)v$ *if we can maintain at least* v *neurons with the highest gradient contribution, thus ensuring the convergence of soft-training.*

Proof: In soft-training, we select partial neurons to join each training cycle for acceleration. Denote the neurons' gradient in soft-training as $ST(g(\Theta_t))$ and simplify $g_t(\Theta_t)$ as g. Each neuron's g has a probability p_i to be selected in each cycle. Therefore, the gradient variance could be reformulated as:

$$ST(g) = \left[D_1\frac{g_1}{p_1}, D_2\frac{g_2}{p_2}, \cdots, D_d\frac{g_d}{p_d}\right], \quad (5)$$

where $D_i \in \{0,1\}$ is the mask indicating the selection results of each neuron, and each gradient term g_i divided by probability p_i denotes the unbiased estimation of g_i. *Moreover, p_i cannot be 0 therefore each neuron shouldn't be inactivated for a long-term.* Thus, the variance of $ST(g)$ could be then reformulated as:

$$\mathbb{E}\sum_{i=1}^{m}\left[ST(g)_i^2\right] = \sum_{i=1}^{m}\left[\frac{g_i^2}{p_i^2}\times p_i + 0\times(1-p_i)\right] = \sum_{i=1}^{m}\frac{g_i^2}{p_i}, \quad (6)$$

which shows the variance of gradient is related to the probability p_i. Therefore, the trade-off between p_i and gradient variance could be formulated as an optimization problem:

$$\min_{p}\sum_{i=1}^{m} p_i \quad \text{s.t.} \quad \sum_{i=1}^{m}\frac{g_i^2}{p_i} \le (1+\epsilon)\sum_{i=1}^{m} g_i^2, \quad (7)$$

where ϵ controls the variance of gradient.

Condition of Convergence. The convergence of soft-training could then be ensured with the following neuron selection criteria. During each soft-training cycle, we keep the P_sn_c neurons with highest convergence contributions joining the next training cycle. Let C represents the set of such high contribution neurons and their total number is v, i.e., $p_i = 1$ for $i \le v$. In order to determine the value of v, we have:

$$\sum_{i=1}^{m}\frac{g_i^2}{p_i} - (1+\epsilon)\sum_{i=1}^{m} g_i^2 = \sum_{i=1}^{v} g_{(i)}^2 + \sum_{i=v+1}^{m}\frac{|g_{(i)}|}{\lambda} - (1+\epsilon)\sum_{i=1}^{m} g_i^2 = 0. \quad (8)$$

Thus according to [19], we can guarantee the convergence with v and $\rho = \epsilon$ and $\mathbb{E}\left[\|ST(g)\|_0\right]$ is bounded by:

$$\begin{aligned}\mathbb{E}\left[\|ST(g)\|_0\right] &= \sum_{i=1}^{m} p_i = \sum_{i\in C_v} p_i + \sum_{i\notin C_v} p_i \\ &\le v + \frac{\rho^2 v\left\|g_{C_v}\right\|_2^2}{\rho\left\|g_{C_v}\right\|_2^2 + (1+\rho)\left\|g_{C_v}\right\|_2^2} \le (1+\rho)v.\end{aligned} \quad (9)$$

Applying the aforementioned dynamic neuron selection, the gradients in the soft-training will be bounded by Eq. 9 and thus soft-training convergence could be achieved.

VI. Optimizations for Soft-Training

In this section, we propose several schemes to enhance overall performance based on the algorithm analysis and improve collaboration scalability.

TABLE I: 4 Stragglers with Heterogeneous Resource

Constraints	Nano (CPU)	Raspberry	Deeplen (GPU)	DeepLen (CPU)
Comp. W (GFLOPS)	7	6	5.5	4.5
Mem. U (MB)	252	150	100	110
Tim. C (Mins)	20.6	23.8	27.2	34

Comp. W: Computation Workload; Mem. U: Memory Usage; Tim. C: Time cost;

A. *Neuron Rotation Regulation*

According to the proposition-2 and the corresponding condition, the gradient variance of soft-training is bounded by P_s, which is the ratio of neurons with the highest contribution U^{ij}. If $P_s = 1$, ρ will equal 0, which means the local training with full neurons on a normal device. According to the previous works [18], [20], P_s usually can be selected from 0.05 to 0.1 for stragglers during FL process.

Another condition requirement is: neurons should not be inactivated for the long-term, which may introduce 0 for p_i and stale parameter issue. In order to avoid this, we pull the long-term skipped neurons back to training cycles timely. Specifically, during aggregation, each straggler will send its partial updated gradients index to the global device. The global device will record all currently skipped neurons and update their skipped cycles value C_s. In every aggregation cycle, once a certain neuron's C_s exceeds a pre-define threshold (we set as $1 + \frac{m}{\sum p_i n_i}$), the global device will inform the corresponding straggler and the targeted neuron will rejoin the training.

B. *Model Aggregation Optimization with Heterogeneity*

From the perspective of the overall collaboration process, the proposed soft-training can maintain a balanced model updating of the stragglers. However, diving into each parameter aggregation cycle, multiple stragglers will introduce various partial models with diverged structures. Such a limited model size will bring additional errors to the global model. Hence, we introduce heterogeneous model aggregation scheme: according to the updating model size of device, assign each device with different weights to adjust their contributions:

$$\min_{\Theta} \quad f(\Theta) := \frac{1}{N} \sum_{n=1}^{N} \alpha_n f_n(\Theta) \tag{10}$$

where α_n is the adjusting ratio, $\sum \alpha_n = 1$ and $\alpha = r_n / \sum r_n$. r_n is the ratio of neurons selected on each device. A larger α_n means a more complete model structure will have more contributions to the collaboration performance.

C. *Collaboration Scalability Optimization*

During the practical FL process, more devices will dynamically join the collaboration. We leverage the following steps to improve the collaboration scalability of our method: once *Helios* detecting a new device is added into FL, it will first fetch device's hardware computation resources and compare them with the existed stragglers. During this step, both approximation and profiling approaches can be used to identify the new adding device based on whether the FL has enough profiling budgets. If the new device is identified as a straggler, *Helios* will assign it with a pre-defined model volume or adapt its model size based on the resource constraints.

VII. Experiment

A. *Experiment Setup*

Testing Platform Setting: In our experiment, we leverage resource-based profiling to identify the stragglers. We first profile the several edge devices' resource configurations. Then, by adjusting the configuration of CPU/GPU bandwidth and memory availability, we can simulate these edge devices' training performance on multiple Nvidia Jetson Nano development boards and let them as stragglers with different resource capabilities. The details of these straggler settings when running *AlexNet* on *CIFAR-10* are shown in Table. I.

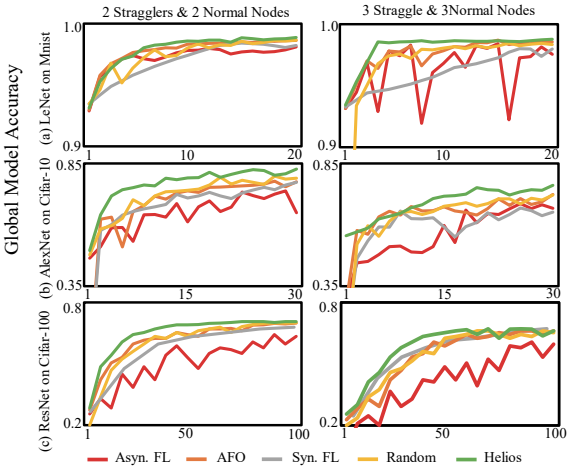

Fig. 5: Soft-training Effectiveness Evaluation

CNN Models and Dataset: In the experiment, three CNN models are used as our testing targets, namely, *LeNet*, *AlexNet*, and *ResNet-18*. The above three CNN models are trained on *MNIST*, *CIFAR-10*, and *CIFAR-100*, respectively.

Baseline Methods: In order to exhibit the superiority of *Helios*, we re-implement five other FL schemes for comparison: (1) *Synchronized FL (Syn. FL)*: All devices (including stragglers) update their parameters synchronously. (2) *Asynchronous FL (Asyn. FL)* Normal edge devices update their parameters immediately after local training without waiting for stragglers. (3) *Random* [12]: In each training, the stragglers randomly select partial model with the expected model structure volume. (4) *AFO* [6]: A optimized asynchronous method aiming to reduce staleness issue of stragglers.

B. *General Helios Performance Evaluation*

We evaluate *Helios* in terms of the converge accuracy and speed by comparing the above baselines in this part. As shown in Table. I, there are two straggler settings involved: (1) Four devices join in the FL with two capable devices and two straggled devices as Strag. 1 and Strag. 2. (2) Six devices join in the FL with three capable devices and three straggled devices as Strag. 1, Strag. 2, and Strag. 3.

Accuracy Evaluation.: Fig. 5 illustrates the experimental results on converge accuracy. The X-axis represents the aggregation cycles of capable edge devices. We can find that *Asyn. FL* always achieves the lowest accuracy due to the information degradation. As for *Syn. FL*, since its training cycle is determined by the stragglers, it shows a much lower aggregation speed than other methods. *Helios* always shows the best accuracy compared to all the baseline methods. Specifically, for *LeNet*, *AlexNet*, and *ResNet18*, *Helios* achieves at most 0.24% to 4.64% accuracy improvement on two kinds of straggler settings.

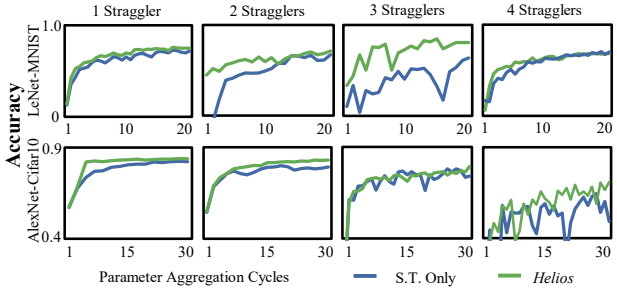

Fig. 6: Model Aggregation Optimization Evaluation

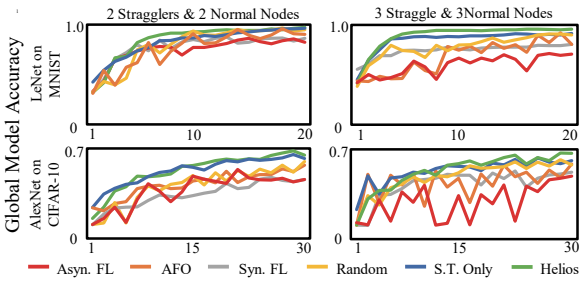

Fig. 7: *Helios* Evaluation with Non-IID Data

Convergence Speed Evaluation: It is clear that *Asyn. FL* has the worst converge speed, and even fail to converge due to the staleness parameters and information loss. Although *AFO* optimizes the asynchronous updates, its convergence is still affected by the staleness parameters of stragglers. On the contrary, *Helios* shows the fastest converge speed, especially for the FL with more stragglers. For the experiments of 6 nodes on *MNIST*, *CIFAR-10*, and *CIFAR-100*, *Helios* approaches convergence after 4, 12, and 40 aggregation cycles respectively. On the contrary, other methods need at least 10, 18, and 50 aggregation cycles. Overall, *Helios* can achieve at most 2.5× speedup than the state-of-the-art methods.

C. *Model Aggregation Optimization Evaluation*

In this part, the effectiveness of model aggregation optimization scheme is also evaluated under the same setting (stragglers are increasing from 1 to 4). We use soft-training without aggregation optimization as the baseline, which is represented as *S.T. Only* . Fig. 6 illustrates the comparison results between *Helios* and *S.T. Only*. We can see that *Helios* achieves at most 17.37% accuracy improvement and reduce the accuracy variance caused by partial model aggregation while accuracy curve of *S.T. Only* still has obvious fluctuations.

D. *Non-IID Setting Evaluation*

In this part, we evaluate the effectiveness of *Helios* with a Non-IID setting (though our proposed technique is more oriented by computational heterogeneity rather than data heterogeneity). Specifically, Non-IID capability is evaluated with 4 and 6 edge devices with 2 and 3 stragglers, respectively. We use the same Non-IID data generation method in [1]. Fig. 7 shows the evaluation results. We can find that Non-IID data brings performance degradation for all methods. However, compared with other methods, *Helios* can always obtain a better converge accuracy and speed.

VIII. Conclusion

In this work, we proposed *Helios* — a heterogeneous-aware FL framework with dynamically balanced collaboration. Leveraging on-device model training consumption profiling and the innovative "soft-training" training scheme, *Helios* can introduce effective local CNN model optimization into FL to eliminate stragglers, while maintaining expected collaborative convergence across all edge devices. Experiments demonstrated that the proposed *Helios* achieves superior training accuracy, speed, as well as Non-IID setting resistance. *Helios* significantly enhanced the applicability and performance of federated learning for training-on-edge.

Acknowledgment: This work was supported in part by NSF CNS-2003211.

References

[1] Y. Zhao and *et al.*, "Federated Learning with Non-IID Data," *arXiv:1806.00582*.

[2] W. Wu and *et al.*, "SAFA: a Semi-Asynchronous Protocol for Fast Federated Learning with Low Overhead," *arXiv:1910.01355*.

[3] Y. Chen and *et al.*, "Asynchronous Online Federated Learning for Edge Devices," *arXiv:1911.02134*.

[4] Y. Chen, X. Sun, and Y. Jin, "Communication-efficient federated deep learning with asynchronous model update and temporally weighted aggregation," *arXiv preprint arXiv:1903.07424*, 2019.

[5] U. Mohammad and S. Sorour, "Adaptive task allocation for asynchronous federated mobile edge learning," *arXiv preprint arXiv:1905.01656*, 2019.

[6] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," *arXiv preprint arXiv:1903.03934*, 2019.

[7] I. Hakimi, S. Barkai, M. Gabel, and A. Schuster, "Taming momentum in a distributed asynchronous environment," *arXiv preprint arXiv:1907.11612*, 2019.

[8] W. Zhang, S. Gupta, X. Lian, and J. Liu, "Staleness-aware async-sgd for distributed deep learning," *arXiv preprint arXiv:1511.05950*, 2015.

[9] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Stand-alone and federated learning under passive and active white-box inference attacks," *arXiv preprint arXiv:1812.00910*, 2018.

[10] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays, "Applied federated learning: Improving google keyboard query suggestions," *arXiv preprint arXiv:1812.02903*, 2018.

[11] T. Nishio and *et al.*, "Client Selection for Federated Learning with Heterogeneous Resources in Mobile Edge," in *ICC'19*.

[12] S. Caldas and *et al.*, "Expanding the Reach of Federated Learning by Reducing Client Resource Requirements," *arXiv:1812.07210*, 2018.

[13] S. Wang and *et al.*, "Adaptive Federated Learning in Resource Constrained Edge Computing Systems," *IEEE Journal on Selected Areas in Communications'19*.

[14] Y. Jiang, S. Wang, B. J. Ko, W.-H. Lee, and L. Tassiulas, "Model pruning enables efficient federated learning on edge devices," *arXiv preprint arXiv:1909.12326*, 2019.

[15] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data," *arXiv preprint arXiv:1811.11479*, 2018.

[16] B. Yuan and *et al.*, "Distributed learning of deep neural networks using independent subnet training," *arXiv preprint arXiv:1910.02120*, 2019.

[17] Z. Xu, F. Yu, Z. Qin, C. Liu, and X. Chen, "Directx: Dynamic resource-aware cnn reconfiguration framework for real-time mobile applications," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2020.

[18] D. Alistarh, T. Hoefler, M. Johansson, N. Konstantinov, S. Khirirat, and C. Renggli, "The convergence of sparsified gradient methods," in *Advances in Neural Information Processing Systems*, 2018, pp. 5973–5983.

[19] J. Wangni, J. Wang, J. Liu, and T. Zhang, "Gradient sparsification for communication-efficient distributed optimization," in *Advances in Neural Information Processing Systems*, 2018, pp. 1299–1309.

[20] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," *arXiv preprint arXiv:1712.01887*, 2017.