Straggler-Resilient Federated Learning: Leveraging the Interplay Between Statistical Accuracy and System Heterogeneity

Amirhossein Reisizadeh [1] **Isidoros Tziotis** [2] **Hamed Hassani** [3] **Aryan Mokhtari** [2] **Ramtin Pedarsani** [1]

Abstract

Federated learning is a novel paradigm that involves learning from data samples distributed across a large network of clients while the data remains local. It is, however, known that federated learning is prone to multiple system challenges including system heterogeneity where clients have different computation and communication capabilities. Such heterogeneity in clients' computation speed has a negative effect on the scalability of federated learning algorithms and causes significant slow-down in their runtime due to slow devices (stragglers). In this paper, we propose `FLANP`, a novel straggler-resilient federated learning meta-algorithm that incorporates statistical characteristics of the clients' data to *adaptively* select the clients in order to speed up the learning procedure. The key idea of `FLANP` is to start the training procedure with faster nodes and gradually involve the slower ones in the model training once the statistical accuracy of the current participating nodes' data is reached, while the final model for each stage is used as a warm-start model for the next stage. Our theoretical results characterize the speedup provided by the meta-algorithm `FLANP` in comparison to standard federated benchmarks for strongly convex losses and non-i.i.d. samples. For particular instances, `FLANP` slashes the overall expected runtime by a factor of $\mathcal{O}(\ln(Ns)/\ln(C))$, where C, N and s are number of clusters, number of nodes per cluster and number of samples per node, respectively. In experiments, `FLANP` demonstrates significant speedups in wall-clock time –up to $6\times$– compared to standard federated learning benchmarks.

[1]UC Santa Barbara [2]The University of Texas at Austin [3]University of Pennsylvania. Correspondence to: Amirhossein Reisizadeh <reisizadeh@ucsb.edu>.

This work was presented at the International Workshop on Federated Learning for User Privacy and Data Confidentiality in Conjunction with ICML 2021 (FL-ICML'21). This workshop does not have official proceedings and this paper is non-archival. Copyright 2021 by the author(s).

1. Introduction

Federated learning is a distributed framework whose objective is to train a model using the data of many clients (nodes), while keeping each node's data local. In contrast with centralized learning, the federated learning architecture allows for preserving the clients' privacy as well as reducing the communication burden caused by transmitting data to a cloud. Nevertheless, as we move towards deploying federated learning in practice, it is becoming apparent that several major challenges still remain and the existing frameworks need to be rethought to address them. Important among these challenges is system (device) heterogeneity due to existence of *straggling nodes* – slow nodes with low computational capability – that significantly slow down the model training [1, 2].

In this paper, we focus on system heterogeneity in federated learning and leverage the interplay between statistical accuracy and system heterogeneity to design a straggler-resilient federated learning method that carefully and adaptively selects a subset of available nodes in each round of training. Federated networks consist of thousands of devices with a wide range of computational, communication, battery power, and storage characteristics. Hence, deploying traditional federated learning algorithms such as `FedAvg` [3] on such a highly heterogeneous cluster of devices results in significant and unexpected delays due to existence of slow clients or stragglers. In most such algorithms, clients participate in the model training regardless of their computational capabilities. Consequently, in each communication round of such methods, the server has to wait for the slowest participating node to complete and upload its local updates which slows down the training process.

In this work, we aim to mitigate the effect of stragglers in federated learning based on an adaptive node participation approach, in which clients are selected to participate in different stages of training according to their computation speed. We call our straggler-resilient scheme a **F**ederated **L**earning method with **A**daptive **N**ode **P**articipation or `FLANP`. The key idea of this scheme is to start the model training procedure with only a few clients which are the fastest among all the nodes. These participating clients continue to train their shared models while interacting with the parameter server. Note that since the server waits only for the participating nodes, it takes a short time for the participating (and fast) clients to promptly train a shared model. This model is, however, not accurate as it is trained over only a small fraction of all the samples. We next increase the number of participating clients and include the next fastest subset of nonparticipating nodes in the training. Note that the model trained from the previous stage can be a warm-start initialization for the current stage.

In a data heterogeneous federated architecture, multiple clusters of clients with different data distributions participate in the model training. Such clustered architecture in federated learning has been the subject of several works [4, 5, 6]. In the proposed `FLANP` algorithm, as time progresses, we gradually increase the number of participating clients in each cluster until we reach the full training set and all clients are involved. Note that in this procedure, the slower clients are only used towards the end of the learning process, where the model is already close to the optimal model of the aggregate loss. Another essential observation is that since the model trained in previous rounds already has a reasonable statistical accuracy and this model serves as the initial point of the next round of the iterative algorithm, the slower nodes of each cluster are only needed to contribute in the final rounds of training, leading to a smaller wall-clock time. This is in contrast with having all nodes participate in training from the beginning, which leads to computation time of each round being determined by the slowest node. In this paper, we formally characterize the speedup obtained by the proposed adaptive node participation scheme when `FedGATE` [7] is used as the optimization subroutine for solving each Empirical Risk Minimization (ERM) problem. We would like to emphasize that `FLANP` is a general meta-algorithm that can be employed with any federated learning subroutine studied in the literature [3, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22]. Next, we state a summary of our main contributions:

- We present a straggler-resilient federated learning framework that leverages the interplay between statistical accuracy and device heterogeneity by adaptively activating heterogeneous clients.
- We specify the proposed meta-algorithm with a federated learning subroutine and present its optimization guarantees for strongly convex risks. Further, we characterize the wall-clock time of the proposed straggler-resilient scheme and demonstrate analytically that it achieves up to $\mathcal{O}(\ln(Ns))$ speedup compared to standard benchmarks, where N and s respectively denote the number of clients per cluster and the number of data samples per client.
- Our numerical experiments demonstrate that our adaptive node participation approach significantly speeds up federated learning benchmarks –with either full or partial node participation– in convex and nonconvex settings.

Related work. System (device) heterogeneity challenge, which refers to the case that clients have different computational, communication and storage characteristics, has been studied in the literature. Asynchronous methods have demonstrated improvements in distributed data centers. However, such methods are less desirable in federated settings as they rely on bounded staleness of slow clients [23, 24]. The active sampling approach is another direction in which the server aims for aggregating as many local updates as possible within a predefined time span [25]. More recently, [26] proposed a normalized averaging method to mitigate stragglers in federated systems and the objective inconsistency due to mismatch in clients' local updates. Deadline-based computation has also been studied to mitigate stragglers in decentralized settings [27].

The idea of adaptive sample size training in which we solve a sequence of geometrically increasing ERM problems has been used previously for solving large-scale ERM problems. In particular, it has been shown that this scheme improves the overall computational cost of both first-order [28, 29] and second-order [30, 31, 32] methods for achieving the statistical accuracy of the full training set. In this paper, we exploit this idea to develop `FLANP` for a different setting to address the issue of device heterogeneity in federated learning.

2. Federated learning setup

We consider a federated architecture where C clusters of clients (nodes) interact with a central server. Nodes are assigned to clusters based on their data distribution, meaning that two nodes belong to the same cluster if their underlying data distributions are the same, and two nodes that belong to different clusters have different data distributions. Hence, for any cluster $c \in [C] := \{1, \cdots, C\}$, we assume that the samples of nodes in cluster c are drawn from a common distribution $\mathcal{D}_c$. We also assume that there are N nodes within each cluster and each of them has access to s samples.[1]

We let $L^{c,i}(\boldsymbol{w})$ represent the empirical risk over the s

[1]For readability, we consider equal N and s across the network. However, our method and results extend to the case that N and s vary across the clusters and clients while remaining within the same order of magnitude. Also, note that we do not need to know the clustering pattern. Indeed, several federated clustering methods have been proposed which one can employ to identify the clustering of clients [4, 5, 6].

samples stored on client $i \in [N]$ in cluster c denoted by $\{z_1^{c,i}, \cdots, z_s^{c,i}\}$ drawn from $\mathcal{D}_c$. That is, $L^{c,i}(\boldsymbol{w}) := \frac{1}{s}\sum_{j=1}^{s} \ell(\boldsymbol{w}, z_j^{c,i})$, where the loss function $\ell(\boldsymbol{w}, z)$ indicates how well the model $\boldsymbol{w}$ performs with respect to the sample z. For an arbitrary $n \in [N]$, define $L_n^c(\boldsymbol{w})$ as the collective empirical risk corresponding to samples of nodes $\{1, \cdots, n\}$ in cluster c, which is formally defined as $L_n^c(\boldsymbol{w}) := \frac{1}{n}\sum_{i=1}^{n} L^{c,i}(\boldsymbol{w})$. Indeed, the collective empirical risk of *all nodes* in cluster c is $L_N^c(\boldsymbol{w})$. Finally, we aggregate all the empirical risks $L_n^c(\boldsymbol{w})$ over all clusters and define the empirical risk

$$L_n(\boldsymbol{w}) := \frac{1}{C}\sum_{c=1}^{C} L_n^c(\boldsymbol{w}). \tag{1}$$

Empirical risk $L_n(\boldsymbol{w})$ captures the aggregate risk corresponding to $C \cdot n \cdot s$ *non-iid* data samples from distributions $\mathcal{D}_1, \cdots, \mathcal{D}_C$. We let $\boldsymbol{w}_n^*$ denote the optimal minimizer of the loss $L_n(\boldsymbol{w})$, i.e., $\boldsymbol{w}_n^* = \arg\min_{\boldsymbol{w}} L_n(\boldsymbol{w})$. Note that the problem of finding a global model for the aggregate loss of all available CN nodes in the network is a special case of (1) for $n = N$, i.e.,

$$\min_{\boldsymbol{w}} L_N(\boldsymbol{w}) = \frac{1}{C}\sum_{c=1}^{C} L_N^c(\boldsymbol{w}) = \frac{1}{CNs}\sum_{c=1}^{C}\sum_{i=1}^{N}\sum_{j=1}^{s} \ell(\boldsymbol{w}, z_j^{c,i}). \tag{2}$$

Also, note that empirical risk of cluster c denoted by $L_N^c(\boldsymbol{w})$ is a surrogate for the expected risk of that cluster defined as $L^c(\boldsymbol{w}) := \mathbb{E}_{Z\sim\mathcal{D}_c}[\ell(\boldsymbol{w}, Z)]$. Our ultimate goal is to minimize the expected loss $L(\boldsymbol{w})$, defined as the average of expected risks of the clusters in the network, i.e., $\min_{\boldsymbol{w}} L(\boldsymbol{w}) := \frac{1}{C}\sum_{c=1}^{C} L^c(\boldsymbol{w})$. We denote the minimizer of L by $\boldsymbol{w}^*$. Also, distributions $\mathcal{D}_c$ are unknown and we only have access to a finite number of realizations. Hence, we settle to solve problem (2).

Statistical Accuracy. The difference of expected and empirical risks of each cluster c denoted by $L_n^c(\boldsymbol{w}) - L^c(\boldsymbol{w})$ is referred to as estimation error, which approaches zero as the number of samples in the empirical risk becomes larger. In this case, since $L_n^c(\boldsymbol{w})$ captures ns samples, we assume that there exists a constant V_{ns} bounding the estimation error with high probability, $\sup_{\boldsymbol{w}} |L_n^c(\boldsymbol{w}) - L^c(\boldsymbol{w})| \leq V_{ns}$. The estimation error V_{ns} has been deeply studied in the statistical learning literature [33, 34]. It has been shown that for strongly convex functions the estimation error is proportional to the inverse of sample size [35, 36]. In this work, we also assume that $V_{ns} = \frac{v}{ns}$ for a constant v independent of n and s. Note that for the loss function L_n^c, once we find a point $\tilde{\boldsymbol{w}}$ that has an optimization error of V_{ns}, i.e., $L_n^c(\tilde{\boldsymbol{w}}) - L_n^c(\boldsymbol{w}_n^*) \leq V_{ns}$, there is no gain in improving the optimization error as the overall error with respect to the expected risk L^c would not improve. Hence, when we find a point $\tilde{\boldsymbol{w}}$ such that $L_n^c(\tilde{\boldsymbol{w}}) - L_n^c(\boldsymbol{w}_n^*) \leq V_{ns}$, we state that it has reached the statistical accuracy of L_n^c. By the same argument, one can show that the estimation error corresponding to the aggregate losses over all the clusters is at most $\mathcal{O}(\frac{1}{ns})$, i.e., $\sup_{\boldsymbol{w}} |L_n(\boldsymbol{w}) - L(\boldsymbol{w})| \leq \frac{v}{ns}$ w.h.p. Our goal is to find a solution $\boldsymbol{w}_N$ that is within the statistical accuracy of L_N defined in (2).

System heterogeneity model. As mentioned earlier, federated clients attribute a wide range of computational powers leading to significantly different processing times for a fixed computing task such as gradient computation and local model update. To be more specific, for each cluster c and node i in it, we let T_i^c denote the (expected) time to compute one local model update. The time for such update is mostly determined by the computation time of a fixed batch-size stochastic gradient of the local empirical risk $L^{c,i}(\boldsymbol{w})$. Clearly, larger T corresponds to slower clients or stragglers.

3. Adaptive node participation approach

Several federated learning algorithms have been proposed to solve the ERM problem in (2) such as `FedAvg` [3], `FedProx` [37], `SCAFFOLD` [21], `DIANA` [38], and `FedGATE` [7]. These methods consist of several rounds of local computations by the clients and communication with the server. As explained earlier, federated clients operate in a wide range of computational characteristics, and, therefore, the server has to wait for the slowest node in each communication round to complete its local computation task. Since device participation in such methods is irrespective of their computation speeds, the slowest nodes determine the overall runtime which causes significant slow-down.

In this section, we describe our adaptive node participation approach to mitigate stragglers in federated learning, and lay out the intuition behind it. Our proposal, `FLANP`, is essentially a meta-algorithm that can be specified with the choice of any particular federated learning subroutine.

3.1. `FLANP`: A straggler-resilient federated learning meta-algorithm

As discussed in Section 2, we assume that our network consists of C clusters, where each cluster contains N available nodes with possibly different computation times (See footnote 1). Our proposal to address the device heterogeneity and mitigate the stragglers is as follows.

The server first solves the ERM problem corresponding to n_0 fastest nodes in each cluster, where n_0 is much smaller than the total number of available nodes N in each cluster. To identify the n_0 fastest nodes in each cluster, the server first broadcasts a short hand-shake message to all nodes in that cluster and waits for the first n_0 nodes that respond. These n_0C nodes will participate in the training process

Algorithm 1: FLANP meta-algorithm

```
Initialize $n = n_0$ participating nodes per cluster, initial global model $\boldsymbol{w}_{n_0}$
while $n \leq N$ do
    while $L_n(\boldsymbol{w}_n) - L_n(\boldsymbol{w}_n^*) > V_{ns}$ do
        participating nodes $[n]$ from all clusters update local models via Federated_Solver
        server waits for the updates of the first $n$ nodes from each cluster
        server aggregates local models from participants and updates global model $\boldsymbol{w}_n$
    end
    $n \leftarrow \min\{2n, N\}$          % doubling the participants in all clusters
end
```

in the first stage. Using Federated_Solver which is a federated learning subroutine of choice, e.g., FedAvg or FedGATE, the $n_0 C$ participating nodes proceed to minimize the empirical risk corresponding to their samples, which we denote by $L_{n_0}(\boldsymbol{w})$ as defined in (1). This continues until the $n_0 C$ nodes reach their corresponding statistical accuracy, that is, they reach a global model $\boldsymbol{w}_{n_0}$ such that $L_{n_0}(\boldsymbol{w}_{n_0}) - L_{n_0}(\boldsymbol{w}_{n_0}^*) \leq V_{n_0 s}$. Note that at this stage the server has to wait only for the slowest client among the n_0 participating nodes, which is potentially much faster than the slowest node of the network.

Per our discussion in Section 2, a more accurate solution than $\boldsymbol{w}_{n_0}$ would not be beneficial. Therefore, once statistical accuracy of L_{n_0} is achieved, the procedure is terminated and we increase the number of participating nodes in each cluster from n_0 to $2n_0$. To select the $2n_0$ fastest nodes in each cluster, we repeat the hand-shaking communication protocol that we discussed. Then, the selected nodes use Federated_Solver to find the minimizer of the loss corresponding to $2n_0 C$ participating nodes, while using the solution of the previous stage $\boldsymbol{w}_{n_0}$ as their starting point. Note that since the samples of nodes within each cluster come from the same distribution, we can show that the solutions of two successive stages with $n_0 C$ and $2n_0 C$ participants are close to each other (see Section 4).

Again, in this stage, the training process terminates when we find a point $\boldsymbol{w}_{2n_0}$ within the statistical accuracy of the loss corresponding to $2n_0 C$ participating nodes, i.e., $L_{2n_0}(\boldsymbol{w}_{2n_0}) - L_{2n_0}(\boldsymbol{w}_{2n_0}^*) \leq V_{2n_0 s}$. In the stage with $2n_0 C$ participating nodes, the computation delay is determined by the slowest participating node among $2n_0 C$ nodes, which is slower than the previous stage with $n_0 C$ participating nodes, but still faster than the slowest node of the network. The procedure of geometrically increasing the number of participating nodes continues till the set of participating nodes contains all the available N nodes in each cluster, and these nodes find the final global model $\boldsymbol{w}_N$ within the statistical accuracy of the global loss function $L_N(\boldsymbol{w})$. Algorithm 1 summarizes the straggler-resilient meta-algorithm.

Remark 1 *In our proposed scheme, clients' computation speeds are not needed, and the parameter server figures out the fastest n nodes only by following the presented handshaking protocol.*

From a high-level perspective, Algorithm 1 exploits faster nodes in the beginning of the learning procedure to promptly reach a global model withing their statistical accuracy. By doing so, the server avoids waiting for slower nodes to complete their local updates; however, the optimality gap of such models are relatively large since only a small fraction of data samples have contributed in the global model. By gradually increasing the number of participating nodes and activating slower nodes, the quality of the global model improves while the synchronous computation slows down due to slower nodes. The key point is that slower nodes join the learning process towards the end.

The criterion $L_n(\boldsymbol{w}_n) - L_n(\boldsymbol{w}_n^*) > V_{ns}$ in Algorithm 1 verifies that the current global model satisfies the statistical accuracy corresponding to n participating nodes $\{1, \cdots, n\}$. This condition, however, is not easy to check since the optimal solution $\boldsymbol{w}_n^*$ is unknown. A sufficient and computationally feasible criterion is to check if $\|\nabla L_n(\boldsymbol{w}_n)\|^2 \leq 2\mu V_{ns}$, when ℓ is μ-strongly convex.

3.2. FLANP via FedGATE

As FLANP in Algorithm 1 is a general mechanism to mitigate stragglers in federated settings, one needs to specify the inner optimization subroutine Federated_Solver to quantify the speedup of the proposed approach. This subroutine could be any federated learning algorithm, but here we focus on FedGATE [7], a federated algorithm that employs gradient tracking to provide tight convergence guarantees for nodes with heterogeneous data distributions.

Why FedGATE*?* FLANP is a meta-procedure that can be used for *any* federated learning solver other than FedGATE to make it resilient against straggling nodes. Nevertheless, we use FedGATE as the subroutine since it can handle the case that local gradients are not an unbiased estimator of the global loss gradient, which is the case in our setting. Algorithm 2 demonstrates how adaptive node participation in FLANP is adopted to mitigate straggler delays in FedGATE. Let us briefly discuss the main points of Algorithm 2 and defer the implementation details to the appendix. Here, the gradient tracking variables $\delta_{c,i}$ aim to correct the directions of local updates at node i from cluster c by tracking the difference of local gradients $\widetilde{\nabla} L^{c,i}$ and global gradients ∇L_n

Algorithm 2: FLANP via FedGATE

```
Initialize $n = n_0$ participating nodes per cluster, initial
  model $\boldsymbol{w}_{n_0}$, initial gradient tracking $\delta_{c,i}^{(0)} = 0$ for
  participating nodes $i \in [n_0]$ in all clusters $c \in [C]$
while $n \leq N$ do
    $r = 0$    % reset round counter for each stage
    for participating nodes $i \in [n]$ and all clusters $c \in [C]$ do
        $\delta_{c,i}^{(0)} = 0$    % reset gradient tracking
    end
    while $\|\nabla L_n(\boldsymbol{w}_n)\|^2 > 2\mu V_{ns}$ do
        for participating nodes $i \in [n]$ in all clusters $c \in [C]$ do
            $\boldsymbol{w}_{c,i}^{(0,r)} = \boldsymbol{w}_n$
            for $t = 0, \cdots, \tau_n - 1$ do
                set $d_{c,i}^{(t,r)} = \widetilde{\nabla} L^{c,i}(\boldsymbol{w}_{c,i}^{(t,r)}) - \delta_{c,i}^{(r)}$ and
                  update $\boldsymbol{w}_{c,i}^{(t+1,r)} = \boldsymbol{w}_{c,i}^{(t,r)} - \eta_n d_{c,i}^{(t,r)}$
            end
            upload $\Delta_{c,i}^{(r)} = (\boldsymbol{w}_n - \boldsymbol{w}_{c,i}^{(\tau_n,r)})/\eta_n$ and
              update $\delta_{c,i}^{(r+1)} = \delta_{c,i}^{(r)} + \frac{1}{\tau_n}(\Delta_{c,i}^{(r)} - \Delta^{(r)})$
        end
        server broadcasts $\Delta^{(r)} = \frac{1}{nC}\sum_{c=1}^{C}\sum_{i=1}^{n}\Delta_{c,i}^{(r)}$
          and $\boldsymbol{w}_n \leftarrow \boldsymbol{w}_n - \eta_n\gamma_n\Delta^{(r)}$
        participating nodes $i \in [n], c \in [C]$ upload
          gradients $\nabla L^{c,i}(\boldsymbol{w}_n)$ to server, $r \leftarrow r + 1$
    end
    $n \leftarrow \min\{2n, N\}$    % doubling the participants
end
```

such that directions $d_{c,i}$ closely follow the correct global gradient direction.

4. Theoretical results

Next, we analyze FLANP outlined in Algorithm 2, which employs FedGATE as its subroutine. We first characterize optimization guarantees of Algorithm 2. Using such results, we derive the expected runtime of our proposed algorithm and the speedup it provides compared to naive methods.

Connection between two successive stages. As we discussed in Section 3.1, we expect the solution of each stage with m participating nodes to be close to the solution of the next stage with n nodes, where $n > m$, if the larger set of nodes contain the smaller set. This is due to the fact that within each cluster, samples are drawn from the same distribution. To formalize this claim, consider a subset of m participating nodes and a model $\boldsymbol{w}_m^*$ within their statistical accuracy, i.e., $L_m(\boldsymbol{w}_m) - L_m(\boldsymbol{w}_m^*) \leq V_{ms}$. Next, we show that the suboptimality error of $\boldsymbol{w}_m$ for the next loss with n nodes is small, when the set of n nodes contains m nodes.

Proposition 1 *Consider two subsets of nodes $\mathcal{N}_m \subseteq \mathcal{N}_n$ and assume that model $\boldsymbol{w}_m$ attains the statistical accuracy for the empirical risk associated with nodes in $\mathcal{N}_m$, i.e., $\|\nabla L_m(\boldsymbol{w}_m)\|^2 \leq 2\mu V_{ms}$ where the loss function ℓ is μ-strongly convex. Then the suboptimality of $\boldsymbol{w}_m$ for risk L_n is w.h.p. bounded above by $L_n(\boldsymbol{w}_m) - L_n(\boldsymbol{w}_n^*) \leq \frac{2(n-m)}{n}(V_{(n-m)s} + V_{ms}) + V_{ms}$.*

Proposition 1 demonstrates that a model attaining the statistical accuracy for m nodes can be used as an initial model for the ERM corresponding to a larger set with n nodes. In particular, when the number of participating nodes is doubled, i.e., $n = 2m$, then the initial sub-optimality error is bounded above by $L_n(\boldsymbol{w}_m) - L_n(\boldsymbol{w}_n^*) \leq 3V_{ms}$.

Optimization guarantees. Next, we characterize the required communication and computation for solving each subproblem. Specifically, consider the case that we are given a model $\boldsymbol{w}_m$ which is within the statistical accuracy of L_m corresponding to m fastest nodes in each cluster, and the goal is to find a new model $\boldsymbol{w}_n$ that is within the statistical accuracy of L_n corresponding to n fastest nodes of each cluster, where $n = 2m$. To analyze this procedure, we must specify three parameters: the choice of stepsizes η_n, γ_n, the number of local updates τ_n at each participating node, and the number of communication rounds with the server R_n. For these parameters we use index n, as they refer to the case that n nodes participate in the training. Next, we state our main assumptions.

Assumption 1 *The loss $\ell(\boldsymbol{w}, z)$ is μ-strongly convex with respect to $\boldsymbol{w}$, and the gradient $\nabla\ell(\boldsymbol{w}, z)$ is L-Lipschitz continuous. The condition number is defined as $\kappa := L/\mu$.*

The conditions in Assumption 1 imply that the empirical risks $L_n(\boldsymbol{w})$ and local loss functions $L^{c,i}(\boldsymbol{w})$ are μ-strongly convex and have L-Lipschitz gradients. As we discussed in Section 2, the gap between the expected and the empirical risks corresponding to nCs data samples can be bounded as $|L_n(\boldsymbol{w}) - L(\boldsymbol{w})| \leq V_{ns}$, with high probability. Next, we formalize this assumption.

Assumption 2 *The approximation error for the expected loss $L(\boldsymbol{w})$ using nCs samples of n nodes per cluster in the empirical risk $L_n(\boldsymbol{w})$ is w.h.p. upper-bounded as $\sup_{\boldsymbol{w}} |L_n(\boldsymbol{w}) - L(\boldsymbol{w})| \leq V_{ns}$, where $V_{ns} = \mathcal{O}(1/ns)$. Moreover, we assume that the approximation error for gradients is upper-bounded by $\sup_{\boldsymbol{w}} \|\nabla L_n(\boldsymbol{w}) - \nabla L(\boldsymbol{w})\| \leq \sqrt{V_{ns}}$, w.h.p. We also assume that the diversity of the gradients $\nabla L^c(\boldsymbol{w})$ is bounded as $\frac{1}{C}\sum_{c=1}^{C}\|\nabla L^c(\boldsymbol{w}) - \nabla L(\boldsymbol{w})\|^2 \leq \rho$, for any $\boldsymbol{w}$ and some constant ρ.*

Theorem 1 *Consider the federated ERM problem in (2) and suppose Assumptions 1 and 2 hold. Let* `FLANP` *in Algorithm 2 be initialized with the fastest n_0 nodes in each cluster and the model $\boldsymbol{w}_{n_0}$. Moreover, suppose the variance of stochastic local gradients is bounded as $\mathbb{E}[\|\widetilde{\nabla} L^{c,i}(\boldsymbol{w}) - \nabla L^{c,i}(\boldsymbol{w})\|^2] \leq \sigma^2$ for all nodes i and clusters c. At any stage of Algorithm 2 with nC participating nodes, if for sufficiently small α_n the stepsizes are $\eta_n = \alpha_n/\tau_n\sqrt{n}$, $\gamma_n = \sqrt{n}/2\alpha_n L$, and each node runs $\tau_n = 1.5s\sigma^2/v$ local updates, then nodes reach the statistical accuracy of L_n after $R_n = 12\kappa \ln(6)$ rounds of communication. Here v captures the constant term in the statistical accuracy $V_{ns} = \frac{v}{ns}$.*

The result in Theorem 1 guarantees that if we initialize Algorithm 2 with n_0 fastest nodes of each cluster and in each stage the participating nodes update their local models according to Algorithm 2 for $\tau = \mathcal{O}(s)$ iterations and $R = \mathcal{O}(\kappa)$ rounds, before doubling the number of participating nodes, then at the end of the final stage in which all N nodes of each cluster are participating, we reach a model $\boldsymbol{w}_N$ that attains the statistical accuracy of the empirical risk $L_N(\boldsymbol{w})$. Specifically, we have $\mathbb{E}[L_N(\boldsymbol{w}_N) - L_N(\boldsymbol{w}_N^*)] \leq V_{Ns}$. Note that to obtain the best guarantee, τ_n and R_n are independent of number of participating nodes n, while the stepsizes η_n and γ_n change as the number of participating nodes increases.

Wall-clock time analysis. We have thus far established the convergence properties of Algorithm 2. It is, however, equally important to show that it provably mitigates stragglers in a federated learning framework and hence speeds up the overall wall-clock time. In the following, we first characterize the run-time of Algorithm 2 and then compare it with the one for straggler-prone `FedGATE` benchmarks.

Let $T_1^c \leq \cdots \leq T_N^c$ denote the computation times of the N available nodes in cluster c. We also denote by $\bar{T}_{\texttt{FLANP}}$ the average runtime of `FLANP` in Algorithm 2 to reach the overall statistical accuracy of the ERM problem corresponding to all nodes defined in (2). As discussed before, at the stage of `FLANP` with n participating node per cluster, the slowest node in each cluster determines the computation time of that stage. More precisely, the computation time of each iteration of `FLANP` with n participating node per cluster is $T_n \coloneqq \max\{T_n^1, \cdots, T_n^C\}$. Since each stage consists of R communication rounds each with τ local updates, the average run-time of each stage is $R\tau T_n$. Therefore, the overall wall-clock time of Algorithm 2 is on average $\bar{T}_{\texttt{FLANP}} = R\tau(T_{n_0} + T_{2n_0} + \cdots + T_N)$ with $R = 12\kappa\ln(6)$ and $\tau = 1.5s\sigma^2/v$ as characterized in Theorem 1.

This further demonstrates how the adaptive node participation approach incorporates faster nodes in order to save in the overall wall-clock time. As Theorem 1 shows, it suffices for each participating node in the straggler-resilient Algorithm 2 to run $R = \mathcal{O}(\kappa)$ rounds of local updates and $\tau = \mathcal{O}(s)$ iterations per round to reach the final statistical accuracy. Therefore, the overall wall-clock time of Algorithm 2 is order-wise $\bar{T}_{\texttt{FLANP}} = \mathcal{O}(\kappa s\sigma^2(T_{n_0} + T_{2n_0} + \cdots + T_N))$.

To quantify the speedup provided by our proposed method, we need to characterize the wall-clock time for the non-adaptive benchmark `FedGATE`. Note that in this benchmark, all the N available nodes are participating in the training process from the beginning.

Proposition 2 *The average runtime for the non-adaptive benchmark* `FedGATE` *to solve the federated ERM problem (2) and to reach the statistical accuracy of all the samples of the N nodes is $\bar{T}_{\texttt{FedGATE}} = \mathcal{O}(\kappa s\sigma^2 \ln(Ns)T_N)$ where T_N is the unit computation time of the slowest node.*

As expected, the result in Proposition 2 indicates that as all N nodes in each cluster participate in training since the beginning of the algorithm, the overall wall-clock time depends only on the slowest node across all clusters with computation time $T_N = \max\{T_N^1, \cdots, T_N^C\}$.

Thus far, we have characterized the order-wise expressions of the average wall-clock time for `FLANP` and `FedGATE` methods as follows

$$\bar{T}_{\texttt{FLANP}} = \mathcal{O}(\kappa s\sigma^2\left(T_{n_0} + T_{2n_0} + \cdots + T_N\right)),$$
$$\bar{T}_{\texttt{FedGATE}} = \mathcal{O}(\kappa s\sigma^2 \ln(Ns)T_N). \tag{3}$$

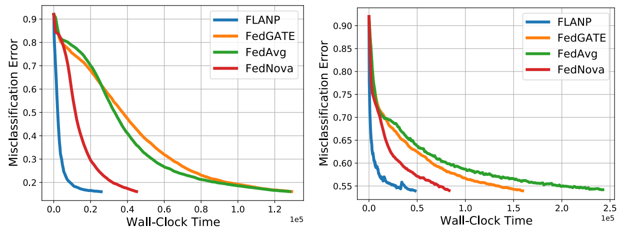
To establish the speedup for the straggler-resilient method, we consider a random exponential time model for clients computation times, which has been widely used to capture the computation delay for distributed clusters [39, 40]. We assume that nodes computation time are independent realizations of an exponential random variable and characterize the speedup of the resilient Algorithm 2 compared to the benchmark `FedGATE`.

Theorem 2 *Let Assumptions 1 and 2 hold and the random computation times be drawn as $T_i^c \sim \exp(\lambda)$ for all clients $i \in [N]$ and clusters $c \in [C]$. Then, the speedup of the proposed* `FLANP` *compared to the naive federated learning method* `FedGATE` *is*

$$\frac{\mathbb{E}[\bar{T}_{\texttt{FLANP}}]}{\mathbb{E}[\bar{T}_{\texttt{FedGATE}}]} \leq \mathcal{O}\left(\frac{1 + \ln(C)}{\ln(Ns)}\right).$$

Theorem 2 establishes $\mathcal{O}(\ln(Ns)/\ln(C))$ speedup for `FLANP` compared to its non-adaptive and straggler-prone benchmark `FedGATE`, when the clients' computation time are drawn from a random exponential time model. Note that this is a significant gain as the number of samples per cluster Ns is often much larger than the number of clusters C.

We have so far considered device heterogeneous clients with potentially well-spread computation speeds and demonstrated the speedup obtained by adaptive node participation approach, particularly in Theorem 2. We would like to add that our method provides provable speedups even for device *homogeneous* clients with identical speeds, implying $T_1 = \cdots = T_N$. Comparing the average runtimes in (3) yields that `FLANP` in Algorithm 2 slashes the expected wall-clock time of `FedGATE` by a factor $\ln(Ns)/\ln(N)$. This observation demonstrates that the adaptive node participation approach results in two different speedups: (i) leveraging faster nodes to speedup the training and (ii) adaptively increase the effective sample size by participating more clients.



(a) MNIST (b) CIFAR10

Figure 1: NN training on $C = 3$ clusters.

5. Numerical experiments and discussion

We conduct various numerical experiments for convex and nonconvex risks and evaluate the performance of the proposed method versus other benchmarks. Below is a brief description for multiple federated benchmarks that we use to compare with the proposed `FLANP` in Algorithm 2.

Benchmarks. `FedAvg` [3]: Nodes update their local model using SGD for τ local iterations in each round. `FedGATE` [7]: This is the subroutine used in Algorithm 2. Here we consider it as a benchmark with update rule similar to the subroutine in Algorithm 2. `FedNova` [26]: In each round, each node i updates its local model for τ_i iterations where τ_is vary across the nodes. To mitigate the heterogeneity in τ_is, the parameter server aggregates normalized updates (w.r.t. τ_i) and updates the global model.

We compare the performance of `FLANP` with such benchmarks in terms of communication rounds and wall-clock time –under both full and partial node participation scenarios and highlight its practicality and compatibility. We conduct two sets of experiments on networks consisting of $C = 3$ and $C = 1$ clusters.

Case I: $C = 3$. We partition the available $N = 60$ nodes into $C = 3$ clusters of size 20 and assign their data samples as follows [21]. For each node, we pick $s\%$ of its total samples to be i.i.d., i.e. $s\%$ similarity of samples across all the nodes. For the remaining $(100 - s)\%$, we create heterogeneous samples as follows [41]. For each cluster $c \in \{1, 2, 3\}$, we let the vector $\mathbf{p}^c$ denote the prior distribution of classes for nodes in cluster c. For MNIST [42] $(60,000$ training, $10,000$ test samples) and CIFAR10 [43] $(50,000$ training, $10,000$ test samples) datasets with 10 total classes, $\mathbf{p}^c \in \mathbb{R}^{10}_{\geq 0}$ and $\|\mathbf{p}^c\|_1 = 1$. Then, for each node $i \in \{1, \cdots, 20\}$ in cluster c, we draw a random vector $\mathbf{q}^{c,i} \sim \mathsf{Dir}(\alpha \mathbf{p}^c)$ from a Dirichlet distribution with α a constant, which determines the portions of the 10 classes for this node. In our experiments with MNIST and CIFAR10, we consider three different classes with equal weights as the prior distribution for each of the $C = 3$ clusters and use the one class left out as part of the i.i.d. samples pool. Note that after realizing $\mathbf{q}^{c,i}$, each node is assigned samples from all the 10 classes but with biased portions.
Next, we use the clustering method proposed in [4] to cluster the $N = 60$ nodes into $C = 3$ clusters. With some parameter tuning, it recovers the ground truth clustering for all the 60 nodes. After we obtain the clustering, we run `FLANP` using `FedGATE` as the federated solver along with other benchmarks. As Figure 1 shows, the adaptive node participation approach leads `FLANP` to speedups of up to $6\times$ compared to `FedGATE`.

Case II: $C = 1$. In this case, we consider several scenarios as follows.

Uniform computation speeds. The network consists of N nodes where we realize and then fix the computation speed of each node from $[50, 500]$ uniformly at random.

Logistic Regression. In a network of $N = 50$ nodes, each client stores $s = 1200$ samples of MNIST. As demonstrated in Figure 2 (left), `FLANP` is slightly outperformed by `FedGATE` at the initial rounds. This is however expected as `FLANP` starts with only a fraction of nodes which leads to less accurate models. With respect to wall-clock time however, `FLANP` outperforms both `FedAvg` and `FedGATE` benchmarks due to the fact that the initial participating nodes are indeed the fastest ones. As Figure 2 (right) shows, `FLANP` significantly speeds up the training by up to $2.1\times$ compared to `FedGATE`. We defer our experiments for linear regression to the appendix.

Neural Network Training. We train a fully connected neural network with two hidden layers with 128 and 64 neurons and compare with three other benchmarks including `FedNova` which is stragglers-resilient. We conduct two sets of experiments over CIFAR10 (and MNIST in the appendix) on a network of $N = 20$ clients. Figure 3 demonstrates that `FLANP` significantly accelerates the training by up to $3\times$ and $4\times$ compared to `FedNova`.

Random exponential computation speeds. We conduct another set of experiments using the same setup described above. However, we here pick the clients' computation speed to be i.i.d. random exponential variables, i.e. consistent with Theorem 2. We train a fully connected neural network with two hidden layers with 128 and 64 neurons on

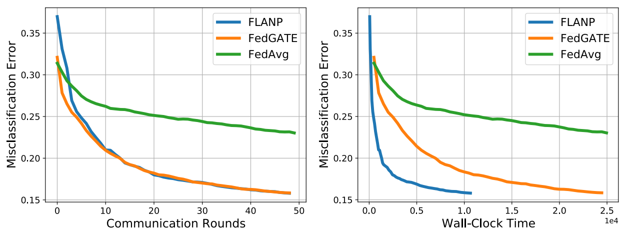


Figure 2: Logistic Regression over MNIST

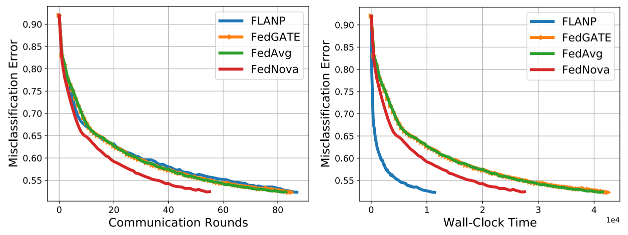


Figure 3: NN on CIFAR10 (random uniform speeds)

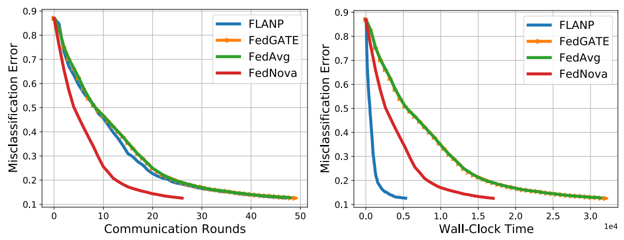
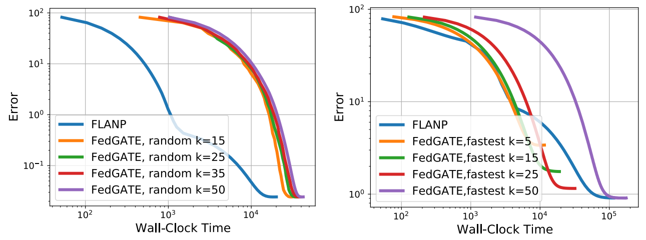


Figure 4: NN on MNIST (random exponential speeds)



(a) k nodes are randomly picked. (b) k fastest nodes are picked.

Figure 5: Partial node participation

MNIST as demonstrated in Figure 4.

Comparison with partial node participation methods. Thus far, we have compared FLANP with federated benchmarks in which *all* of the available nodes participate in every round. To demonstrate the resiliency of FLANP to partial node participation methods, we consider two different scenarios. First, we compare the wall-clock time of a linear regression model with Gaussian synthetic samples using FLANP with partial node participation FedGATE in which only k out of $N = 50$ nodes are randomly picked and participate in each round. As demonstrated in Figure 5(a), FLANP is significantly faster than FedGATE with partial node participation. Second, we consider the case that the k participating nodes are not randomly picked, rather are the fastest clients. As shown in Figure 5(b), although partial participation methods with k fastest nodes begin to outperform FLANP, towards the end of the training, they suffer from higher training error saturation as the data samples of *only* k nodes contribute in the trained model and hence the final model is significantly inaccurate.

FLANP with other federated solvers. To illustrate the compatibility of FLANP with solvers other than FedGATE, we train the neural network on MNIST and employ FedAvg and FedNova as solvers of FLANP. As shown in Figure 6, FLANP is able to significantly speedup all three solvers.

Lastly, we note that from the practical point of view, there are several heuristic approaches to estimate the constant parameters μ, v, V_{ns} in Algorithm 2. We conducted an experiment to learn a linear regression model with Gaussian synthetic data in which none of the constants are assumed to be known. Rather, we heuristically tune the threshold for each stage transition (i.e. doubling the nodes) by monitoring the norm of the global gradient and successively halving the threshold. As shown in Figure 7, the performance of such heuristic methods is indeed close to FLANP which highlights its practicality.

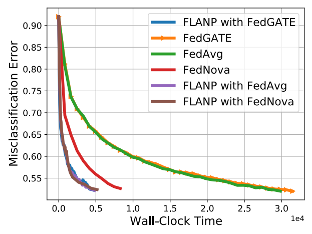


Figure 6: FLANP with other federated solvers

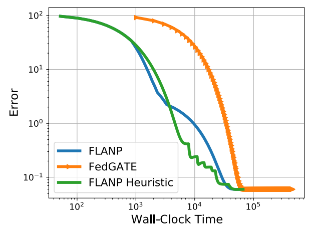


Figure 7: FLANP with heuri-stic parameter tuning

References

[1] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *arXiv preprint arXiv:1908.07873*, 2019.

[2] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, *et al.*, "Advances and open problems in federated learning," *arXiv preprint arXiv:1912.04977*, 2019.

[3] 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*, pp. 1273–1282, PMLR, 2017.

[4] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[5] A. Ghosh, J. Hong, D. Yin, and K. Ramchandran, "Robust federated learning in a heterogeneous environment," *arXiv preprint arXiv:1906.06629*, 2019.

[6] D. K. Dennis, T. Li, and V. Smith, "Heterogeneity for the win: One-shot federated clustering," *arXiv preprint arXiv:2103.00697*, 2021.

[7] F. Haddadpour, M. M. Kamani, A. Mokhtari, and M. Mahdavi, "Federated learning with compression: Unified analysis and sharp guarantees," *arXiv preprint arXiv:2007.01154*, 2020.

[8] J. Wang and G. Joshi, "Cooperative sgd: A unified framework for the design and analysis of communication-efficient sgd algorithms," *arXiv preprint arXiv:1808.07576*, 2018.

[9] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," in *International Conference on Learning Representations*, 2019.

[10] Z. Huo, Q. Yang, B. Gu, L. C. Huang, *et al.*, "Faster on-device training using new federated momentum algorithm," *arXiv preprint arXiv:2002.02090*, 2020.

[11] G. Malinovsky, D. Kovalev, E. Gasanov, L. Condat, and P. Richtarik, "From local sgd to local fixed point methods for federated learning," *arXiv preprint arXiv:2004.01442*, 2020.

[12] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[13] F. Zhou and G. Cong, "On the convergence properties of a k-step averaging stochastic gradient descent algorithm for nonconvex optimization," *arXiv preprint arXiv:1708.01012*, 2017.

[14] F. Haddadpour and M. Mahdavi, "On the convergence of local descent methods in federated learning," *arXiv preprint arXiv:1910.14425*, 2019.

[15] F. Haddadpour, M. M. Kamani, M. Mahdavi, and V. Cadambe, "Local sgd with periodic averaging: Tighter analysis and adaptive synchronization," in *Advances in Neural Information Processing Systems*, pp. 11082–11094, 2019.

[16] A. K. R. Bayoumi, K. Mishchenko, and P. Richtarik, "Tighter theory for local sgd on identical and heterogeneous data," in *International Conference on Artificial Intelligence and Statistics*, pp. 4519–4529, 2020.

[17] S. U. Stich and S. P. Karimireddy, "The error-feedback framework: Better rates for sgd with delayed gradients and compressed communication," *arXiv preprint arXiv:1909.05350*, 2019.

[18] J. Wang and G. Joshi, "Adaptive communication strategies to achieve the best error-runtime trade-off in local-update sgd," *arXiv preprint arXiv:1810.08313*, 2018.

[19] A. Koloskova, N. Loizou, S. Boreiri, M. Jaggi, and S. U. Stich, "A unified theory of decentralized sgd with changing topology and local updates," *arXiv preprint arXiv:2003.10422*, 2020.

[20] X. Liang, S. Shen, J. Liu, Z. Pan, E. Chen, and Y. Cheng, "Variance reduced local sgd with lower communication complexity," *arXiv preprint arXiv:1912.12844*, 2019.

[21] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for on-device federated learning," *arXiv preprint arXiv:1910.06378*, 2019.

[22] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečnỳ, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," *arXiv preprint arXiv:2003.00295*, 2020.

[23] S. U. Stich, "Local sgd converges fast and communicates little," in *ICLR 2019 ICLR 2019 International Conference on Learning Representations*, no. CONF, 2019.

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

[25] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1–7, IEEE, 2019.

[26] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," *arXiv preprint arXiv:2007.07481*, 2020.

[27] A. Reisizadeh, H. Taheri, A. Mokhtari, H. Hassani, and R. Pedarsani, "Robust and communication-efficient collaborative learning," in *Advances in Neural Information Processing Systems*, pp. 8388–8399, 2019.

[28] A. Mokhtari and A. Ribeiro, "First-order adaptive sample size methods to reduce complexity of empirical risk minimization," in *NeurIPS*, 2017.

[29] A. Mokhtari, A. Ozdaglar, and A. Jadbabaie, "Efficient nonconvex empirical risk minimization via adaptive sample size methods," in *AISTATS*, 2019.

[30] A. Mokhtari, H. Daneshmand, A. Lucchi, T. Hofmann, and A. Ribeiro, "Adaptive Newton method for empirical risk minimization to statistical accuracy," in *NeurIPS*, 2016.

[31] M. Eisen, A. Mokhtari, and A. Ribeiro, "Large scale empirical risk minimization via truncated adaptive Newton method," in *AISTATS*, 2018.

[32] M. Jahani, X. He, C. Ma, A. Mokhtari, D. Mudigere, A. Ribeiro, and M. Takác, "Efficient distributed hessian free algorithm for large-scale empirical risk minimization via accumulating sample strategy," in *AISTATS*, 2020.

[33] V. Vapnik, *The nature of statistical learning theory*. Springer science & business media, 2013.

[34] O. Bousquet, *Concentration inequalities and empirical processes theory applied to the analysis of learning algorithms*. PhD thesis, École Polytechnique: Department of Applied Mathematics Paris, France, 2002.

[35] P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe, "Convexity, classification, and risk bounds," *Journal of the American Statistical Association*, vol. 101, no. 473, pp. 138–156, 2006.

[36] R. Frostig, R. Ge, S. M. Kakade, and A. Sidford, "Competing with the empirical risk minimizer in a single pass," in *Conference on learning theory*, pp. 728–763, 2015.

[37] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *arXiv preprint arXiv:1812.06127*, 2018.

[38] K. Mishchenko, E. Gorbunov, M. Takáč, and P. Richtárik, "Distributed learning with compressed gradient differences," *arXiv preprint arXiv:1901.09269*, 2019.

[39] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1514–1529, 2017.

[40] A. Reisizadeh, S. Prakash, R. Pedarsani, and A. S. Avestimehr, "Coded computation over heterogeneous clusters," *IEEE Transactions on Information Theory*, vol. 65, no. 7, pp. 4227–4242, 2019.

[41] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," *arXiv preprint arXiv:1909.06335*, 2019.

[42] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

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

A. Proof of Proposition 1

Let us present and prove the following lemma which includes the claim in Proposition 1.

Lemma 1 *Consider two subsets of nodes $\mathcal{N}_m \subseteq \mathcal{N}_n$ and assume that model $\boldsymbol{w}_m$ attains the statistical accuracy for the empirical risk associated with nodes in $\mathcal{N}_m$, that is, $\|\nabla L_m(\boldsymbol{w}_m)\|^2 \leq 2\mu V_{ms}$ where the loss function ℓ is μ-strongly convex. Then the suboptimality of $\boldsymbol{w}_m$ for risk L_n, i.e., $L_n(\boldsymbol{w}_m) - L_n(\boldsymbol{w}_n^*)$ is w.h.p. bounded above as follows:*

$$L_n(\boldsymbol{w}_m) - L_n(\boldsymbol{w}_n^*) \leq \frac{2(n-m)}{n}\left(V_{(n-m)s} + V_{ms}\right) + V_{ms}. \tag{4}$$

Moreover, norms of the global gradient is upper-bounded w.h.p. as follows:

$$\|\nabla L_n(\boldsymbol{w}_m)\|^2 \leq 2\left(\frac{n-m}{n}\right)^2\left(V_{(n-m)s}^{1/2} + V_{ms}^{1/2}\right)^2 + 4\mu V_{ms} \tag{5}$$

Furthermore, for any $i \in [N]$ we have

$$\frac{1}{C}\sum_{c=1}^{C}\left\|\nabla L^{c,i}(\boldsymbol{w}_m)\right\|^2 \leq 4(2\mu+1)V_{ms} + 4V_s + 4\rho, \tag{6}$$

where we assume that the diversity of the gradients $\nabla L^c(\boldsymbol{w})$ is bounded as

$$\frac{1}{C}\sum_{c=1}^{C}\left\|\nabla L^c(\boldsymbol{w}) - \nabla L(\boldsymbol{w})\right\|^2 \leq \rho, \tag{7}$$

for any $\boldsymbol{w}$ and some constant ρ.

Proof. We begin the proof of Lemma 1 by proving the inequality in (4). Let us decompose the sub-otpimiality error $L_n(\boldsymbol{w}_m) - L_n(\boldsymbol{w}_n^*)$ to four difference terms as follows:

$$\begin{aligned}L_n(\boldsymbol{w}_m) - L_n(\boldsymbol{w}_n^*) = {} & L_n(\boldsymbol{w}_m) - L_m(\boldsymbol{w}_m) + L_m(\boldsymbol{w}_m) - L_m(\boldsymbol{w}_m^*)\\ & + L_m(\boldsymbol{w}_m^*) - L_m(\boldsymbol{w}_n^*) + L_m(\boldsymbol{w}_n^*) - L_n(\boldsymbol{w}_n^*).\end{aligned} \tag{8}$$

From definition of local empirical risks in (1), the difference of local risks $L_n(\boldsymbol{w})$ and $L_m(\boldsymbol{w})$ for any $\boldsymbol{w}$ can be bounded w.h.p. as follows:

$$\begin{aligned}L_n(\boldsymbol{w}) - L_m(\boldsymbol{w}) \leq & \left|L_n(\boldsymbol{w}) - L_m(\boldsymbol{w})\right|\\ = & \left|\frac{1}{C}\sum_{c=1}^{C} L_n^c(\boldsymbol{w}) - \frac{1}{C}\sum_{c=1}^{C} L_m^c(\boldsymbol{w})\right|\\ \leq & \frac{1}{C}\sum_{c=1}^{C}\left|L_n^c(\boldsymbol{w}) - L_m^c(\boldsymbol{w})\right|.\end{aligned} \tag{9}$$

For any cluster $c \in [C]$, we can bound the difference of local risks $L_n^c(\boldsymbol{w})$ and $L_m^c(\boldsymbol{w})$ as follows:

$$\begin{aligned}
\left|L_n^c(\boldsymbol{w}) - L_m^c(\boldsymbol{w})\right| &= \left|\frac{1}{n}\sum_{i\in\mathcal{N}_n\setminus\mathcal{N}_m} L^{c,i}(\boldsymbol{w}) - \frac{n-m}{n}\cdot\frac{1}{m}\sum_{i\in\mathcal{N}_m} L^{c,i}(\boldsymbol{w})\right| \\
&= \frac{n-m}{n}\left|\frac{1}{n-m}\sum_{i\in\mathcal{N}_n\setminus\mathcal{N}_m} L^{c,i}(\boldsymbol{w}) - \frac{1}{m}\sum_{i\in\mathcal{N}_m} L^{c,i}(\boldsymbol{w})\right| \\
&\leq \frac{n-m}{n}\left|\frac{1}{n-m}\sum_{i\in\mathcal{N}_n\setminus\mathcal{N}_m} L^{c,i}(\boldsymbol{w}) - L^c(\boldsymbol{w})\right| \\
&\quad + \frac{n-m}{n}\left|\frac{1}{m}\sum_{i\in\mathcal{N}_m} L^{c,i}(\boldsymbol{w}) - L^c(\boldsymbol{w})\right| \\
&\leq \frac{n-m}{n}\left(V_{(n-m)s} + V_{ms}\right), \quad w.h.p., \qquad (10)
\end{aligned}$$

which together with (9) yields that

$$L_n(\boldsymbol{w}) - L_m(\boldsymbol{w}) \leq \frac{n-m}{n}\left(V_{(n-m)s} + V_{ms}\right). \qquad (11)$$

The last inequality in (10) is implied from the statistical accuracy assumption, where the gap between the empirical risk of $(n-m)s$ samples from distribution $\mathcal{D}_c$, i.e. $\frac{1}{n-m}\sum_{i\in\mathcal{N}_n\setminus\mathcal{N}_m} L^{c,i}(\boldsymbol{w})$, and the corresponding expected risk $L^c(\boldsymbol{w}) := \mathbb{E}_{Z\sim\mathcal{D}_c}[\ell(\boldsymbol{w}, Z)]$ can be bounded by $V_{(n-m)s}$. By a similar argument, such gap for ms samples can be bounded by V_{ms}. This also shows how Assumption 2 for aggregated risks over all clusters can be independently derived given the statistical accuracy assumption within each cluster.

We now proceed to bound the next term in (8), that is the optimality gap $L_m(\boldsymbol{w}_m) - L_m(\boldsymbol{w}_m^*)$. Using the strong convexity assumption in Assumption 1 and the condition $\|\nabla L_m(\boldsymbol{w}_m)\|^2 \leq 2\mu V_{ms}$ assumed to hold in the statement of the lemma, we can write

$$L_m(\boldsymbol{w}_m) - L_m(\boldsymbol{w}_m^*) \leq \frac{1}{2\mu}\left\|\nabla L_m(\boldsymbol{w}_m)\right\|^2 \leq \frac{2\mu V_{ms}}{2\mu} = V_{ms}. \qquad (12)$$

Next, the term $L_m(\boldsymbol{w}_m^*) - L_m(\boldsymbol{w}_n^*)$ in (8) can be simply bounded as $L_m(\boldsymbol{w}_m^*) - L_m(\boldsymbol{w}_n^*) \leq 0$, since $\boldsymbol{w}_m^*$ is the minimizer of $L_m(\boldsymbol{w})$. Finally, to bound $L_m(\boldsymbol{w}_n^*) - L_n(\boldsymbol{w}_n^*)$ in (8), we use the result in (11) which holds for any $\boldsymbol{w}$ and here we pick $\boldsymbol{w} = \boldsymbol{w}_n^*$ to conclude

$$L_m(\boldsymbol{w}_n^*) - L_n(\boldsymbol{w}_n^*) \leq \frac{n-m}{n}\left(V_{(n-m)s} + V_{ms}\right). \qquad (13)$$

Putting the upper bounds for the four terms in (8) together proves inequality (4) which is the same claim as in Proposition 1.

Next we prove inequality (5) by first noting the following:

$$\left\|\nabla L_n(\boldsymbol{w}_m)\right\|^2 \leq 2\left\|\nabla L_n(\boldsymbol{w}_m) - \nabla L_m(\boldsymbol{w}_m)\right\|^2 + 2\left\|\nabla L_m(\boldsymbol{w}_m)\right\|^2. \qquad (14)$$

The first term $\|\nabla L_n(\boldsymbol{w}_m) - \nabla L_m(\boldsymbol{w}_m)\|$ can be bounded as follows. For any $\boldsymbol{w}$ we can write

$$\begin{aligned}
\left\|\nabla L_n(\boldsymbol{w}) - \nabla L_m(\boldsymbol{w})\right\| &= \left\|\frac{1}{C}\sum_{c=1}^{C}\nabla L_n^c(\boldsymbol{w}) - \frac{1}{C}\sum_{c=1}^{C}\nabla L_m^c(\boldsymbol{w})\right\| \\
&\leq \frac{1}{C}\sum_{c=1}^{C}\left\|\nabla L_n^c(\boldsymbol{w}) - \nabla L_m^c(\boldsymbol{w})\right\|. \qquad (15)
\end{aligned}$$

For any cluster $c \in [C]$, we can bound the difference $\left\|\nabla L_n^c(\boldsymbol{w}) - \nabla L_m^c(\boldsymbol{w})\right\|$ as follows:

$$\begin{aligned}
\left\|\nabla L_n^c(\boldsymbol{w}) - \nabla L_m^c(\boldsymbol{w})\right\| &= \left\| \frac{1}{n} \sum_{i \in \mathcal{N}_n} \nabla L^{c,i}(\boldsymbol{w}) - \frac{1}{m} \sum_{i \in \mathcal{N}_m} \nabla L^{c,i}(\boldsymbol{w}) \right\| \\
&= \left\| \frac{1}{n} \sum_{i \in \mathcal{N}_n \setminus \mathcal{N}_m} \nabla L^{c,i}(\boldsymbol{w}) - \frac{n-m}{n} \cdot \frac{1}{m} \sum_{i \in \mathcal{N}_m} \nabla L^{c,i}(\boldsymbol{w}) \right\| \\
&= \frac{n-m}{n} \left\| \frac{1}{n-m} \sum_{i \in \mathcal{N}_n \setminus \mathcal{N}_m} \nabla L^{c,i}(\boldsymbol{w}) - \frac{1}{m} \sum_{i \in \mathcal{N}_m} \nabla L^{c,i}(\boldsymbol{w}) \right\| \\
&\le \frac{n-m}{n} \left\| \frac{1}{n-m} \sum_{i \in \mathcal{N}_n \setminus \mathcal{N}_m} \nabla L^{c,i}(\boldsymbol{w}) - \nabla L^c(\boldsymbol{w}) \right\| \\
&\quad + \frac{n-m}{n} \left\| \frac{1}{m} \sum_{i \in \mathcal{N}_m} \nabla L^{c,i}(\boldsymbol{w}) - \nabla L^c(\boldsymbol{w}) \right\| \\
&\le \frac{n-m}{n} \left(V_{(n-m)s}^{1/2} + V_{ms}^{1/2} \right), \quad w.h.p., \qquad (16)
\end{aligned}$$

which together with (15) yields that

$$\left\|\nabla L_n(\boldsymbol{w}) - \nabla L_m(\boldsymbol{w})\right\| \le \frac{n-m}{n} \left(V_{(n-m)s}^{1/2} + V_{ms}^{1/2} \right), \quad w.h.p. \qquad (17)$$

In the last inequality in (16), we used the assumption that within any cluster c, the difference of empirical risk gradient for $(n-m)s$ samples from distribution $\mathcal{D}_c$ and the corresponding expected risk gradient $\nabla L^c(\boldsymbol{w})$ can be bounded by $V_{(n-m)s}$. Similarly, such gap for ms samples from $\mathcal{D}_c$ can be bounded by V_{ms}. Assumption 2 states a more general account of such assumption for the aggregated risk across any number of clusters. Here, we basically showed how Assumption 2 can be derived if we assume such upper-bound holds within any cluster. Together with (14) and the assumption of the lemma, that is $\left\|\nabla L_m(\boldsymbol{w}_m)\right\|^2 \le 2\mu V_{ms}$, the claim in (5) is concluded:

$$\|\nabla L_n(\boldsymbol{w}_m)\|^2 \le 2 \left(\frac{n-m}{n}\right)^2 \left(V_{(n-m)s}^{1/2} + V_{ms}^{1/2} \right)^2 + 4\mu V_{ms}. \qquad (18)$$

Finally, we prove the claim in inequality (6) by bounding node i's local gradient $\nabla L^i(\boldsymbol{w}_m)$ as follows:

$$\begin{aligned}
\left\|\nabla L^{c,i}(\boldsymbol{w}_m)\right\|^2 &\le 4\left\|\nabla L^{c,i}(\boldsymbol{w}_m) - \nabla L^c(\boldsymbol{w}_m)\right\|^2 + 4\left\|\nabla L^c(\boldsymbol{w}_m) - \nabla L(\boldsymbol{w}_m)\right\|^2 \\
&\quad + 4\left\|\nabla L_m(\boldsymbol{w}_m) - \nabla L(\boldsymbol{w}_m)\right\|^2 + 4\left\|\nabla L_m(\boldsymbol{w}_m)\right\|^2 \\
&\le 4V_s + 4\left\|\nabla L^c(\boldsymbol{w}_m) - \nabla L(\boldsymbol{w}_m)\right\|^2 + 4V_{ms} + 6\mu V_{ms} \\
&= 4(2\mu+1)V_{ms} + 4V_s + 4\left\|\nabla L^c(\boldsymbol{w}_m) - \nabla L(\boldsymbol{w}_m)\right\|^2, \qquad (19)
\end{aligned}$$

where we used Assumption 2 to upper-bound the approximation error of empirical gradients for node i in cluster c with s samples and m nodes with ms samples. Averaging both sides of (19) over $c \in \{1, \cdots, C\}$ yields that

$$\frac{1}{C} \sum_{c=1}^{C} \left\|\nabla L^{c,i}(\boldsymbol{w}_m)\right\|^2 \le 4(2\mu+1)V_{ms} + 4V_s + 4\rho, \qquad (20)$$

where we use the bounded gradient diversity assumption that for any $\boldsymbol{w}$.

B. Proof of Theorem 1

Consider a stage of Algorithm 2 running with n participating nodes. More precisely, n nodes in $\{1, \cdots, n\}$ begin a sequence of local and global model updates according to `FedGATE` initialized with $\boldsymbol{w}_m$ obtained from the previous stage ($n = 2m$).

After R_n communication rounds each with τ_n local updates, the final sub-optimality error is upper-bounded as follows: (refer to Algorithm 2 and Theorem E.6 in [7] with no quantization)

$$\begin{aligned}\mathbb{E}[L_n(\boldsymbol{w}) - L_n(\boldsymbol{w}_n^*)] \leq & \left(1 - \frac{1}{3}\mu\eta_n\gamma_n\tau_n\right)^{R_n} \left(L_n(\boldsymbol{w}_m) - L_n(\boldsymbol{w}_n^*)\right) \\ & + 24\kappa^3 L\tau_n^2\eta_n^2 \frac{1}{Cn}\sum_{c=1}^{C}\sum_{i=1}^{n}\left\|\nabla L^{c,i}(\boldsymbol{w}_m)\right\|^2 + 24\kappa L\tau_n^2\eta_n^2\left\|\nabla L_n(\boldsymbol{w}_m)\right\|^2 \\ & + 24\kappa^2 L^2\tau_n^2\eta_n^3\sigma^2 + 15\kappa L^3\tau_n^3\eta_n^2(\eta_n\gamma_n)^2\frac{\sigma^2}{Cn} + \frac{L}{2}\eta_n\gamma_n\frac{\sigma^2}{Cn}, \end{aligned} \tag{21}$$

where two stepsizes η_n, γ_n satisfy the following conditions:

$$1 - L\eta_n\gamma_n\tau_n + \frac{10\eta_n^2\tau_n^4L^4(\eta_n\gamma_n)^2}{1-\mu\tau_n\gamma_n\eta_n + 20\mu\gamma_n\eta_n^3L^2\eta_n^3} \leq 1 \qquad \& \qquad 30\eta_n^2L^2\tau_n^2 \leq 1. \tag{22}$$

To satisfy the two conditions in (22), we can pick stepsizes η_n, γ_n such that

$$2\eta_n\gamma_n\tau_nL = 1 \qquad \& \qquad 30\eta_n^2L^2\tau_n^2 \leq 1. \tag{23}$$

Now we use the result in Lemma 1 and put $n = 2m$ to conclude that

$$\begin{aligned} L_n(\boldsymbol{w}_m) - L_n(\boldsymbol{w}_n^*) &\leq 3V_{ms}, \\ \|\nabla L_n(\boldsymbol{w}_m)\|^2 &\leq 2(2\mu+1)V_{ms}, \\ \frac{1}{C}\sum_{c=1}^{C}\|\nabla L^{c,i}(\boldsymbol{w}_m)\|^2 &\leq 4(2\mu+1)V_{ms} + 4V_s + 4\rho. \end{aligned} \tag{24}$$

Substituting the three inequalities (24) in the sub-optimality error (21) yields that

$$\begin{aligned}\mathbb{E}[L_n(\boldsymbol{w}) - L_n(\boldsymbol{w}_n^*)] \leq\ & 3\left(1 - \frac{1}{3}\mu\eta_n\gamma_n\tau_n\right)^{R_n} V_{ms} \\ & + 96\kappa^3L\tau_n^2\eta_n^2\left((2\mu+1)V_{ms} + V_s + \rho\right) + 48(2\mu+1)\kappa L\tau_n^2\eta_n^2V_{ms} \\ & + 24\kappa^2L^2\tau_n^2\eta_n^3\sigma^2 + 15\kappa L^3\tau_n^3\eta_n^2(\eta_n\gamma_n)^2\frac{\sigma^2}{n} + \frac{L}{2}\eta_n\gamma_n\frac{\sigma^2}{n}. \end{aligned} \tag{25}$$

We use the fact that $2\eta_n\gamma_n\tau_nL = 1$ and rearrange the terms in (25) and rewrite it as follows:

$$\begin{aligned}\mathbb{E}[L_n(\boldsymbol{w}) - L_n(\boldsymbol{w}_n^*)] \leq\ & 3\left(1 - \frac{1}{6\kappa}\right)^{R_n} V_{ms} \\ & + 96\kappa^3L\tau_n^2\eta_n^2(V_s + \rho) + 48(2\kappa^2+1)(2\mu+1)\kappa L\tau_n^2\eta_n^2V_{ms} \\ & + 24\kappa^2L^2\tau_n^2\eta_n^3\sigma^2 + \frac{15}{4}\kappa L\tau_n\eta_n^2\frac{\sigma^2}{Cn} + \frac{L}{2}\eta_n\gamma_n\frac{\sigma^2}{Cn}. \end{aligned} \tag{26}$$

To ensure that a model $\boldsymbol{w} = \boldsymbol{w}_n$ attains the statistical accuracy of $L_n(\boldsymbol{w})$, i.s. $\mathbb{E}[L_n(\boldsymbol{w}_n) - L_n(\boldsymbol{w}_n^*)] \leq V_{ns}$, it suffices to have each of the six terms in RHS of (26) less than or equal to $V_{ns}/6$. That is,

$$\begin{aligned}
3\left(1-\frac{1}{6\kappa}\right)^{R_n} V_{ms} &\leq \frac{V_{ns}}{6},\\
96\kappa^3 L\tau_n^2\eta_n^2 V_s &\leq \frac{V_{ns}}{12},\\
96\kappa^3 L\tau_n^2\eta_n^2 \rho &\leq \frac{V_{ns}}{12},\\
48(2\kappa^2+1)(2\mu+1)\kappa L\tau_n^2\eta_n^2 V_{ms} &\leq \frac{V_{ns}}{6},\\
24\kappa^2 L^2\tau_n^2\eta_n^3\sigma^2 &\leq \frac{V_{ns}}{6},\\
\frac{15}{4}\kappa L\tau_n\eta_n^2\frac{\sigma^2}{Cn} &\leq \frac{V_{ns}}{6},\\
\frac{L}{2}\eta_n\gamma_n\frac{\sigma^2}{Cn} &\leq \frac{V_{ns}}{6},
\end{aligned} \tag{27}$$

where $n = 2m$ and $V_{ns} = \frac{v}{ns}$ for any n. One can check that the following picks for the stepsizes η_n, γ_n satisfies all the conditions in (22) and (27):

$$\begin{aligned}
\eta_n &= \frac{\alpha_n}{\tau_n\sqrt{n}},\\
\gamma_n &= \frac{\sqrt{n}}{2\alpha_n L},
\end{aligned} \tag{28}$$

where

$$\begin{aligned}
\alpha_n \leq \min\Bigg\{&\frac{1}{24\sqrt{2}\kappa\sqrt{\kappa L}}, \frac{\sqrt{v}}{24\sqrt{2}\kappa\sqrt{\kappa L s\rho}},\\
&\frac{\sqrt{n}}{12\sqrt{2(3\kappa^2+2)(2\mu+1)\kappa L}}, \left(\frac{\sqrt{n}}{96\kappa^2L^2}\right)^{1/3}, \frac{\sqrt{Cnv}}{\sqrt{15\kappa L}}, \frac{\sqrt{n}}{L\sqrt{30}}\Bigg\}.
\end{aligned} \tag{29}$$

Moreover, the first and the last conditions in (27) yield that the number of local updates and the number of communication rounds for the stage with n participating nodes are

$$\begin{aligned}
\tau_n &= \frac{3}{2}\frac{\sigma^2 s}{Cv},\\
R_n &= 12\kappa\ln(6).
\end{aligned} \tag{30}$$

C. Proof of Proposition 2

In order to characterize the runtime of `FedGATE`, we first need to determine its two major parameters τ and R. More precisely, we run `FedGATE` algorithm with all the N available nodes while initialized with arbitrary model $\boldsymbol{w}_0$ and look for τ, R after which the global model $\tilde{\boldsymbol{w}}$ attains the statistical accuracy of $L_N(\boldsymbol{w})$, i.e. $\mathbb{E}[L_N(\tilde{\boldsymbol{w}}) - L_N(\boldsymbol{w}_N^*)] \leq V_{Ns}$. We use the convergence guarantee of `FedGATE` [7] in (21) with $n = N$ nodes, that is,

$$\begin{aligned}
\mathbb{E}[L_N(\boldsymbol{w}) - L_N(\boldsymbol{w}_N^*)] \leq &\left(1-\frac{1}{3}\mu\eta\gamma\tau\right)^R \left(L_N(\boldsymbol{w}_0) - L_N(\boldsymbol{w}_N^*)\right)\\
&+ 24\kappa^3L\tau^2\eta^2\frac{1}{CN}\sum_{c=1}^{C}\sum_{i=1}^{N}\left\|\nabla L^{c,i}(\boldsymbol{w}_0)\right\|^2 + 24\kappa L\tau^2\eta^2\left\|\nabla L_N(\boldsymbol{w}_0)\right\|^2\\
&+ 24\kappa^2L^2\tau^2\eta^3\sigma^2 + 15\kappa L^3\tau^3\eta^2(\eta\gamma)^2\frac{\sigma^2}{CN} + \frac{L}{2}\eta\gamma\frac{\sigma^2}{CN},
\end{aligned} \tag{31}$$

where the stepsizes η, γ satisfy the following conditions:

$$1 - L\eta\gamma\tau + \frac{10\eta^2\tau^4L^4(\eta\gamma)^2}{1-\mu\tau\gamma\eta + 20\mu\gamma\eta^3L^2\eta^3} \leq 1 \qquad \& \qquad 30\eta^2L^2\tau^2 \leq 1. \tag{32}$$

Note that the initial model $\boldsymbol{w}_0$ is arbitrary and therefore the initial sub-optimality error can be treated as a constant (and not scaling with N), that is, $L_N(\boldsymbol{w}_0) - L_N(\boldsymbol{w}_N^*) = \Delta_0$ for a constant $\Delta_0 = \mathcal{O}(1)$. Similarly, we can assume that $\frac{1}{CN}\sum_{c=1}^{C}\sum_{i=1}^{N}\left\|\nabla L^{c,i}(\boldsymbol{w}_0)\right\|^2 = \Delta_0'$ for a constant $\Delta_0' = \mathcal{O}(1)$ which also yields that $\left\|\nabla L_N(\boldsymbol{w}_0)\right\|^2 \leq \Delta_0'$. We can therefore further simplify (31) and write

$$\begin{aligned}\mathbb{E}[L_N(\boldsymbol{w}) - L_N(\boldsymbol{w}_N^*)] \leq \left(1 - \frac{1}{3}\mu\eta\gamma\tau\right)^R \Delta_0 + 24\kappa(\kappa^2+1)L\tau^2\eta^2\Delta_0' \\ + 24\kappa^2L^2\tau^2\eta^3\sigma^2 + 15\kappa L^3\tau^3\eta^2(\eta\gamma)^2\frac{\sigma^2}{N} + \frac{L}{2}\eta\gamma\frac{\sigma^2}{N}.\end{aligned} \tag{33}$$

We furthermore pick the parameters such that $2\eta\gamma\tau L = 1$ which further simplifies (33) as follows:

$$\begin{aligned}\mathbb{E}[L_N(\boldsymbol{w}) - L_N(\boldsymbol{w}_N^*)] \leq \left(1 - \frac{1}{6\kappa}\right)^R \Delta_0 + 24\kappa(\kappa^2+1)L\tau^2\eta^2\Delta_0' \\ + 24\kappa^2L^2\tau^2\eta^3\sigma^2 + \frac{15}{4}\kappa L\tau\eta^2\frac{\sigma^2}{CN} + \frac{L}{2}\eta\gamma\frac{\sigma^2}{CN}.\end{aligned} \tag{34}$$

Now to ensure that $\mathbb{E}[L_N(\tilde{\boldsymbol{w}}) - L_N(\boldsymbol{w}_N^*)] \leq V_{Ns}$ holds for a model $\tilde{\boldsymbol{w}}$ in (34), it suffices to satisfy the following inequalities:

$$\begin{aligned}\left(1 - \frac{1}{6\kappa}\right)^R \Delta_0 &\leq \frac{V_{Ns}}{5}, \\ 24\kappa(\kappa^2+1)L\tau^2\eta^2\Delta_0' &\leq \frac{V_{Ns}}{5}, \\ 24\kappa^2L^2\tau^2\eta^3\sigma^2 &\leq \frac{V_{Ns}}{5}, \\ \frac{15}{4}\kappa L\tau\eta^2\frac{\sigma^2}{CN} &\leq \frac{V_{Ns}}{5}, \\ \frac{L}{2}\eta\gamma\frac{\sigma^2}{CN} &\leq \frac{V_{Ns}}{5},\end{aligned} \tag{35}$$

with $V_{Ns} = \frac{v}{Ns}$. The following picks for the stepsizes satisfy the aforementioned conditions

$$\begin{aligned}\eta &= \frac{\alpha}{\tau\sqrt{Ns}}, \\ \gamma &= \frac{\sqrt{Ns}}{2\alpha L},\end{aligned} \tag{36}$$

where

$$\alpha \leq \min\left\{\frac{\sqrt{c}}{2\sqrt{30}\sqrt{\kappa(\kappa^2+1)L\Delta_0'}}, \left(\frac{\sqrt{Ns}}{96\kappa^2L^2}\right)^{1/3}, \frac{\sqrt{Ns}}{\sqrt{15\kappa L}}, \frac{\sqrt{Ns}}{L\sqrt{30}}\right\}. \tag{37}$$

Moreover, the number of local updates and the number of communication rounds to reach the final statistical accuracy are as follows:

$$\begin{aligned}\tau &= \frac{5}{4}\frac{\sigma^2 s}{Cv} = \mathcal{O}(s), \\ R &= 6\kappa\ln\left(\frac{5\Delta_0 Ns}{v}\right) = \mathcal{O}(\kappa\ln(Ns)).\end{aligned} \tag{38}$$

$$\begin{array}{ccccccc}
T^1_{(1)} & \leq \cdots \leq & T^1_{(n)} & \leq \cdots \leq & T^1_{(N)} \\
\vdots & & \vdots & & \vdots \\
T^c_{(1)} & \leq \cdots \leq & T^c_{(n)} & \leq \cdots \leq & T^c_{(N)} \\
\vdots & & \vdots & & \vdots \\
T^C_{(1)} & \leq \cdots \leq & T^C_{(n)} & \leq \cdots \leq & T^C_{(N)} \\
\max = T_1 & & \max = T_n & & \max = T_N
\end{array}$$

Figure 8: Computation time notation

Now note that the expected runtime of each communication round of `FedGATE` is τT_N as the server has to wait for the slowest node that is node N with processing time T_N. Therefore, the total expected wall-clock time of `FedGATE` to reach the final statistical accuracy of all the samples of the N nodes in $L_N(\boldsymbol{w})$ is

$$\bar{T}_{\texttt{FedGATE}} = R\tau T_N = \mathcal{O}(\kappa s \ln(Ns) T_N),$$

as claimed in Proposition 2.

D. Proof of Theorem 2

Recall the result in Proposition 2 and the following discussion in (3). As discussed, the average runtime for the proposed `FLANP` with `FedGATE` in Algorithm 2 is as follows:

$$\bar{T}_{\texttt{FLANP}} = R_{\texttt{FLANP}}\, \tau_{\texttt{FLANP}} \sum_{i = n_0,\, 2n_0,\, 4n_0, \cdots,\, N} T_i = \frac{18\ln(6)}{Cv} \kappa s \sigma^2 \left(T_{n_0} + T_{2n_0} + \cdots + T_N \right), \tag{39}$$

where $R_{\texttt{FLANP}} = 12\kappa \ln(6)$ and $\tau_{\texttt{FLANP}} = 1.5 s\sigma^2 v^{-1} C^{-1}$ per Theorem 1. Moreover, we showed in Proposition 2 that the expected runtime for `FedGATE` is

$$\bar{T}_{\texttt{FedGATE}} = R_{\texttt{FedGATE}}\, \tau_{\texttt{FedGATE}}\, T_N = \frac{15}{2Cv} \kappa s \sigma^2 \ln\left(\frac{5\Delta_0 N s}{v} \right) T_N. \tag{40}$$

In the case that clients' computation times T_i^cs are random, the expected runtimes are

$$\begin{aligned}
\mathbb{E}[\bar{T}_{\texttt{FLANP}}] &= \frac{18\ln(6)}{Cv} \kappa s \sigma^2 \left(\mathbb{E}[T_{n_0}] + \mathbb{E}[T_{2n_0}] + \cdots + \mathbb{E}[T_N] \right), \\
\mathbb{E}[\bar{T}_{\texttt{FedGATE}}] &= \frac{15}{2Cv} \kappa s \sigma^2 \ln\left(\frac{5\Delta_0 N s}{c} \right) \mathbb{E}[T_N].
\end{aligned} \tag{41}$$

Therefore, in order to derive the runtime gain $\frac{\mathbb{E}[\bar{T}_{\texttt{FLANP}}]}{\mathbb{E}[\bar{T}_{\texttt{FedGATE}}]}$, we first characterize the ratio

$$\frac{\mathbb{E}[T_{n_0}] + \mathbb{E}[T_{2n_0}] + \cdots + \mathbb{E}[T_N]}{\mathbb{E}[T_N]}, \tag{42}$$

where the clients runtimes T_i^c are i.i.d. with random exponential distribution $\exp(\lambda)$ with rate λ.

Before moving further, let us recall and reset our notation. We denote by T_i^c the computation time of node i in cluster c, where $i \in [N]$ and $c \in [C]$. Let us sort the computation times within each cluster and denote them as $T^c_{(1)} \leq T^c_{(2)} \leq T^c_{(3)} \leq \cdots T^c_{(N)}$ for any cluster c. We also denote

$$T_n := \max\{T^1_{(n)}, \cdots, T^C_{(n)}\}, \tag{43}$$

for any $n \in [N]$ (See Figure 8). Without loss of generality and for simplification, let us take $n_0 = 1$ and $\lambda = 1$ and proceed to bound the ratio

$$\frac{\mathbb{E}[T_1] + \mathbb{E}[T_2] + \mathbb{E}[T_4] + \cdots + \mathbb{E}[T_N]}{\mathbb{E}[T_N]}. \tag{44}$$

We initialize `FLANP` with $n_0 = 1$ client per cluster. For this stage, the computation time of each iteration of `FLANP` for cluster c is $T^c_{(1)}$. Therefore, the overall wall-clock time to aggregate from all clusters is $T_1 = \max_{c\in[C]} T^c_{(1)}$. Similarly, when n client per cluster participate in `FLANP`, the wall-clock time for each iteration is $T_n = \max_{c\in[C]} T^c_{(n)}$. Therefore, to determine the total expected wall-clock time of `FLANP`, we need to characterize $\mathbb{E}[T_n]$ for any $1 \leq n \leq N$.

Using Jensen's inequality for any $t > 0$ and $1 \leq n \leq N$, we can write

$$\begin{aligned} e^{t\mathbb{E}[T_n]} &\leq \mathbb{E}[e^{tT_n}] \\ &= \mathbb{E}[\max_{i\in[C]} e^{tT^i_{(n)}}] \\ &\leq \sum_{i\in[C]} \mathbb{E}[e^{tT^i_{(n)}}] \\ &= C \cdot \mathbb{E}[e^{tT^1_{(n)}}] \end{aligned} \tag{45}$$

In order to calculate the moment generating function $\mathbb{E}[e^{tT^1_{(n)}}]$, we can write

$$\begin{aligned} \mathbb{E}[e^{tT^1_{(n)}}] &= \mathbb{E}\left[e^{t\left(T^1_{(n)}-T^1_{(n-1)}\right)} \cdot e^{t\left(T^1_{(n-1)}-T^1_{(n-2)}\right)} \cdot \dots \cdot e^{t\left(T^1_{(2)}-T^1_{(1)}\right)} \cdot e^{tT^1_{(1)}}\right] \\ &= \mathbb{E}\left[e^{t\left(T^1_{(n)}-T^1_{(n-1)}\right)}\right] \cdot \mathbb{E}\left[e^{t\left(T^1_{(n-1)}-T^1_{(n-2)}\right)}\right] \cdot \dots \cdot \mathbb{E}\left[e^{t\left(T^1_{(2)}-T^1_{(1)}\right)}\right] \cdot \mathbb{E}\left[e^{tT^1_{(1)}}\right] \\ &= \frac{1}{1-\frac{t}{N-n+1}} \cdot \frac{1}{1-\frac{t}{N-n+2}} \cdot \dots \cdot \frac{1}{1-\frac{t}{N-1}} \cdot \frac{1}{1-\frac{t}{N}}, \end{aligned} \tag{46}$$

for any $0 < t < N - n + 1$. In above, we use few useful facts. First, we note that we can write the nth order statistics of a random exponential as sum of n independent random exponentials as follows:

$$T^1_{(n)} = \left(T^1_{(n)} - T^1_{(n-1)}\right) + \left(T^1_{(n-1)} - T^1_{(n-2)}\right) + \dots + \left(T^1_{(2)} - T^1_{(1)}\right) + T^1_{(1)}, \tag{47}$$

where

$$T^1_{(1)} \sim \exp\left(\frac{1}{N}\right), \quad T^1_{(i)} - T^1_{(i-1)} \sim \exp\left(\frac{1}{N-i+1}\right). \tag{48}$$

We also used the fact that the moment generating function for a random exponential $T \sim \exp(\lambda)$ with parameter λ is

$$\mathbb{E}[e^{tT}] = \frac{1}{1-\lambda t}, \quad t < \frac{1}{\lambda}. \tag{49}$$

Putting all the above derivations together, we have for any $0 < t < N - n + 1$

$$\mathbb{E}[T_n] \leq \frac{1}{t} \ln\left(\frac{C}{\left(1-\frac{t}{N-n+1}\right)\left(1-\frac{t}{N-n+2}\right)\cdots\left(1-\frac{t}{N-1}\right)\left(1-\frac{t}{N}\right)}\right). \tag{50}$$

Let us pick $t = 1$ in the above upper bound and conclude that

$$\mathbb{E}[T_n] \leq \ln\left(\frac{CN}{N-n}\right). \tag{51}$$

Now recall that the expected run-time for `FLANP` is

$$\mathbb{E}[\bar{T}_{\texttt{FLANP}}] = \frac{18\ln(6)}{Cv}\kappa s\sigma^2 \left(\mathbb{E}[T_1] + \mathbb{E}[T_2] + \mathbb{E}[T_4] + \dots + \mathbb{E}[T_N]\right). \tag{52}$$

Using the bound in (51) we can write

$$\begin{aligned}
\mathbb{E}[T_1] &\leq \ln(C) + \ln(N+1) - \ln(N-1), \\
\mathbb{E}[T_2] &\leq \ln(C) + \ln(N+1) - \ln(N-2), \\
\mathbb{E}[T_4] &\leq \ln(C) + \ln(N+1) - \ln(N-4), \\
\mathbb{E}[T_8] &\leq \ln(C) + \ln(N+1) - \ln(N-8), \\
&\vdots \\
\mathbb{E}[T_{N/2}] &\leq \ln(C) + \ln(N+1) - \ln(N/2), \\
\mathbb{E}[T_N] &\leq \ln(N+1) + \gamma, \qquad (53)
\end{aligned}$$

where $\gamma \approx 0.577$ is the Euler-Mascheroni constant and we assumed that N is a power of 2, i.e. $N = 2^K$. Summing up the above inequalities yield that

$$\begin{aligned}
&\mathbb{E}[T_1] + \mathbb{E}[T_2] + \mathbb{E}[T_4] + \cdots + \mathbb{E}[T_N] \\
&\quad \leq (K+1)\ln\left(2^K+1\right) + K\ln(C) + \gamma - \ln\left(\left(2^K-1\right)\left(2^K-2\right)\left(2^K-4\right)\cdots\left(2^{K-1}\right)\right) \\
&\quad \leq (K+1)\ln\left(2^K+1\right) + K\ln(C) + \gamma - (K^2-K)\ln(2) \\
&\quad \leq (K+1)\left(K\ln(2) + \frac{1}{2^K}\right) + K\ln(C) + \gamma - (K^2-K)\ln(2) \\
&\quad = K\left(\ln(C) + 2\ln(2) + \frac{1}{2^K}\right) + \frac{1}{2^K} + \gamma \qquad (54)
\end{aligned}$$

On the other hand, the expected run-time for FedGATE is as follows

$$\mathbb{E}[\bar{T}_{\texttt{FedGATE}}] = \frac{15}{2Cv}\kappa s\sigma^2 \ln\left(\frac{5\Delta_0 N s}{v}\right)\mathbb{E}[T_N]. \qquad (55)$$

Since T_N is the maximum of NC iid random exponentials, we have

$$\mathbb{E}[T_N] \geq \ln(NC) + \gamma = K\ln(2) + \ln(C) + \gamma. \qquad (56)$$

Putting (54) and (56) together, we can bound the following ratio:

$$\begin{aligned}
\frac{\mathbb{E}[T_1] + \mathbb{E}[T_2] + \mathbb{E}[T_4] + \cdots + \mathbb{E}[T_N]}{\mathbb{E}[T_N]} &\leq \frac{K\left(\ln(C) + 2\ln(2) + \frac{1}{2^K}\right) + \frac{1}{2^K} + \gamma}{K\ln(2) + \ln(C) + \gamma} \\
&\leq 2\left(\ln(C) + 1 + \frac{1}{N}\right). \qquad (57)
\end{aligned}$$

Now, we are able to precisely characterize the speedup gain of FLANP compared to FedGATE according to the expressions in (52), (55) and the ratio in (57) to conclude that

$$\begin{aligned}
\frac{\mathbb{E}[\bar{T}_{\texttt{FLANP}}]}{\mathbb{E}[\bar{T}_{\texttt{FedGATE}}]} &= \frac{12\ln(6)}{5\ln\left(5v^{-1}\Delta_0 Ns\right)}\frac{\mathbb{E}[T_1] + \mathbb{E}[T_2] + \mathbb{E}[T_4] + \cdots + \mathbb{E}[T_N]}{\mathbb{E}[T_N]} \\
&\leq \frac{24\ln(6)}{5\ln\left(5v^{-1}\Delta_0 Ns\right)}\left(\ln(C) + 1 + \frac{1}{N}\right) \\
&= \mathcal{O}\left(\frac{1+\ln(C)}{\ln(Ns)}\right), \qquad (58)
\end{aligned}$$

which concludes the theorem.

E. Additional numerical experiments

In the main paper, we demonstrated the performance of FLANP and other benchmarks for a neural network training over CIFAR-10 in Figure 3, where the clients' computation speed are random exponentials. The following plots illustrate the result of a similar experiment over MNIST dataset.

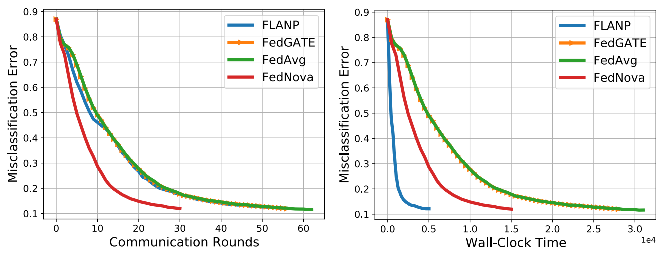


Figure 9: Neural Network Training over MNIST