A. Proof of Theorem 1

A.1. Lemmas For Proving Theorem 1

Before proving Theorem 1, we first provide the following lemmas.

Lemma 1 (One-step SGD)**.** *Assume Assumption 1 and 2. If* $\eta_t \leq \frac{1}{4L}$*, we have*

$$E\left\|\overline{w}_{t+1} - w^\star\right\|^2 \leq (1-\eta_t\mu)E\left\|\overline{w}_t - w^\star\right\|^2 + \eta_t^2 E\left\|\mathbf{e}_t - \overline{\mathbf{e}}_t\right\|^2 + 6L\eta_t^2\Gamma + \frac{2}{K}E\sum_{k=1}^{K}\left\|\overline{w}_t - w_k^t\right\|^2$$

where we define $\mathbf{e}_t = \frac{1}{K}\sum_{k=1}^{K}\nabla F_k(w_t^k, \xi_k)$*,* $\overline{\mathbf{e}}_t = \frac{1}{K}\sum_{k=1}^{K}\nabla F_k(w_t^k)$*,* $\Gamma = F^* - \frac{1}{K}\sum_{k=1}^{K} F_k^\star \geq 0$*.*

Lemma 2 (Bounding the divergence of $\{w_t^k\}$)**.** *Assume Assumption 4, that* η_t *is non-increasing and* $\eta_t \leq 2\eta_{t+E}$ *for all* $t \geq 0$*. It follows that*

$$E\left[\frac{1}{K}\sum_{k=1}^{K}\left\|\overline{w}_t - w_k^t\right\|^2\right] \leq C,$$

where $C = 4\eta_t^2(E-1)^2P^2$.

The proof of Lemma 1 and 2 can be found at [30].

Lemma 3 (Bounding the variance of gradients)**.** *Assume Assumption 3 holds. It follows that*

$$E\left\|\mathbf{e}_t - \overline{\mathbf{e}}_t\right\|^2 \leq \frac{2L_1L_2C}{K} + \frac{1}{K^2}\sum_{k=1}^{K}\epsilon_k^2$$

Proof. According to the dynamic point d, we decompose the stochastic gradients into two sets of gradients regarding the subnets before and after the dynamic point, named as static subnets and dynamic subnets, respectively. We denote the gradients of dynamic subnets as $\mathbf{h}_t$ and $\overline{\mathbf{h}}_t$ and the gradients of static subnets as $\mathbf{g}_t$ and $\overline{\mathbf{g}}_t$ respectively. $\overline{\mathbf{e}}_t = [\overline{\mathbf{g}}_t, \overline{\mathbf{h}}_t]$ and $\mathbf{e}_t = [\mathbf{g}_t, \mathbf{h}_t]$. Then we have:

$$E\left\|\mathbf{e}_t - \overline{\mathbf{e}}_t\right\|^2 \leq E\left\|\mathbf{h}_t - \overline{\mathbf{h}}_t\right\|^2 + E\left\|\mathbf{g}_t - \overline{\mathbf{g}}_t\right\|^2, \tag{7}$$

where $E\left\|\mathbf{h}_t - \overline{\mathbf{h}}_t\right\|^2$ indicates the variance in terms of the dynamic subnets and $E\left\|\mathbf{g}_t - \overline{\mathbf{g}}_t\right\|^2$ indicates the variance in terms of the static subnets.

First, we prove that variance on the dynamic subnets is bounded without the general assumption in federated learning. Through dynamic knowledge propagation, the dynamic subnets are derived from the static subnets in all the clusters. Therefore, we have

$$\mathbf{h}_t = \frac{1}{K}\sum_{j=1}^{K}\nabla H_k(v_t^k, G(u_t^j, \xi_j))$$

and

$$\overline{\mathbf{h}}_t = \frac{1}{K}\sum_{j=1}^{K}\nabla H_k(v_t^k, G(u_t^k, \xi_j)).$$

Then the variance of gradients on the dynamic subnets can be written as

$$\begin{aligned}
E\left\|\mathbf{h}_t - \overline{\mathbf{h}}_t\right\|^2 &= E\left\|\frac{1}{K}\sum_{k=1}^{K}\left(\frac{1}{K}\sum_{j=1}^{K}\nabla H_k(v_t^k, G(u_t^j, \xi_j)) - \sum_{j=1}^{K}\nabla H_k(v_t^k, G(u_t^k, \xi_j))\right)\right\|^2 \\
&= E\left\|\frac{1}{K}\sum_{k=1}^{K}\frac{1}{K}\sum_{j=1}^{K}\left(\nabla H_k(v_t^k, G(u_t^j, \xi_j)) - \nabla H_k(v_t^k, G(u_t^k, \xi_j))\right)\right\|^2 \\
&= \frac{1}{K^4}\sum_{k=1}^{K}\sum_{j=1}^{K}E\left\|\nabla H_k(v_t^k, G(u_t^j, \xi_j)) - \nabla H_k(v_t^k, G(u_t^k, \xi_j))\right\|^2
\end{aligned} \tag{8}$$

Since G is L_2-smooth and ∇G is bounded, G is L_2-Lipschitz (Mean Value Theorem). Given the L_1-smoothness of H_k and the L_2-Lipschitz of G, we have:

$$\begin{aligned}
&\left\|\nabla H_k(v_t^k, G(u_t^j, \xi_j)) - \nabla H_k(v_t^k, G(u_t^k, \xi_j))\right\|^2 && (9)\\
\leq\ & L_1 \left\|G(u_t^j, \xi_j) - G(u_t^k, \xi_j)\right\|^2 \quad (L_1 \text{ smoothness}) \\
\leq\ & L_1 L_2 \left\|u_t^j - u_t^k\right\|^2 \quad (L_2 \text{ Lipschitz}) && (10)
\end{aligned}$$

By combining Eq. 8 and 9, it follows that

$$\begin{aligned}
E\left\|\mathbf{h}_t - \bar{\mathbf{h}}_t\right\|^2 &\leq \frac{1}{K^4} L_1 L_2 \sum_{k=1}^{K} \sum_{j=1}^{K} \left\|u_t^j - u_t^k\right\|^2 \\
&\leq \frac{1}{K^4} L_1 L_2 \sum_{k=1}^{K} \sum_{j=1}^{K} \left(\left\|\overline{u}_t - u_t^k\right\|^2 + \left\|\overline{u}_t - u_t^j\right\|^2\right) \\
&\leq \frac{2L_1 L_2 C}{K^2} \quad (\text{Lemma } 2) && (11)
\end{aligned}$$

Second, we prove that variance on the static subnets is bounded. From Assumption 3, it follows that

$$\begin{aligned}
E\left\|\mathbf{g}_t - \bar{\mathbf{g}}_t\right\|^2 &= E\left\|\frac{1}{K}\sum_{k=1}^{K}\left(\nabla G_k(u_t^k, \xi_t^k) - \nabla G_k(u_t^k)\right)\right\|^2 \\
&= \frac{1}{K^2}\sum_{k=1}^{K} E\left\|\nabla G_k(u_t^k, \xi_t^k) - \nabla G_k(u_t^k)\right\|^2 \\
&\leq \frac{1}{K}\sum_{k=1}^{K} \epsilon_k^2 \quad (Assumption\ 3). && (12)
\end{aligned}$$

By combining Eq. 7, 11, and Eq. 12, we have

$$E\left\|\mathbf{e}_t - \bar{\mathbf{e}}_t\right\|^2 \leq \frac{2L_1 L_2 C}{K} + \frac{1}{K}\sum_{k=1}^{K} \epsilon_k^2.$$

□

A.2. Main Proof of Theorem 1

Proof. From Lemma 1, Lemma 2, and Lemma 3, it follows that

$$E\left\|\overline{w}_{t+1} - w^\star\right\|^2 \leq (1 - \eta_t \mu)\left\|\overline{w}_t - w^\star\right\|^2 + \eta_t^2 B, \tag{13}$$

where

$$B = \frac{1}{K}(2L_1 L_2 C + \sum_{k=1}^{K} \epsilon_k^2) + 6L\Gamma + \frac{2C}{\eta_t^2},$$

$$C = 4\eta_t^2 (E-1)^2 P^2.$$

By the L-smoothness of $F(\cdot)$, it follows that

$$\begin{aligned}
E[F(\bar{w}_t)] - F^* &\leq \frac{L}{2}\left\|\overline{w}_t - w^\star\right\|^2 \\
&\leq \frac{L}{2}\left[(1 - \eta_t \mu)\left\|\overline{w}_{t-1} - w^\star\right\|^2 + \eta_t^2 B\right].
\end{aligned}$$

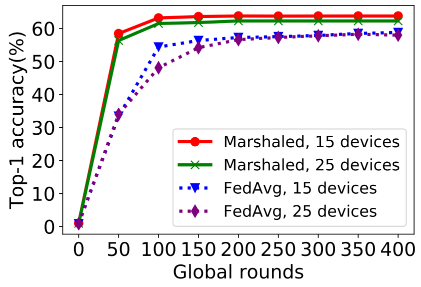

Figure 3. Convergence comparison between Marshaled Learning and FedAvg on the ResNet networks and the non-iid CIFAR-100 dataset.

We denote $\gamma = \max\{8\frac{L}{\mu}, E\} - 1$ and follow the induction proof in [30]. Then it follows

$$E[F(\bar{w}_t)] - F^* \leq \frac{2L}{\mu(\gamma + t)} \left(\frac{B}{\mu} + 2L \left\| w_0 - w^\star \right\|^2 \right).$$

This completes the proof. □

Remark. We compare the convergence between Marshaled Learning and FedAvg. First, compared with FedAvg, Marshaled Learning only requires the assumption of variance bound of gradients on the static subnets (Assumption 3), removing the assumption on the dynamic subnets. Second, Marshaled Learning shows the advantages of convergence rates over FedAvg. In FedAvg, we have to include the assumption on the bound of the variance of stochastic gradients: $E \left\| \nabla F_k(w_t^k, \xi_t^k) - \nabla F_k(w_t^k) \right\|^2 \leq \zeta_k^2$ Comparing the term B (in Eq. 6) between Marshaled Learning and FedAvg, we find that Marshaled Learning achieves faster convergence rate when $\sum_{k=1}^{K} 4\eta_t^2(E-1)^2 P^2 L_1 L_2 + \epsilon_k^2 \leq \zeta_k^2$ holds. Note that the first term in the left is irrelevant to the data distribution over clients and η_t^2 is decreasing over the training, while ζ_k^2 is very large in the high data heterogeneity scenarios. Hence, the inequality holds in most cases, demonstrating the advantage of Marshaled Learning shows.

B. Marshaled Learning Algorithm

Algorithm 1 illustrates Marshaled Learning, including dynamic knowledge propagation and straggler mitigation.

C. Ablation Studies

C.1. Convergence of Marshaled Learning

Fig. 3 shows the convergence of Marshaled Learning and FedAvg on the non-iid data distributions. We observe that the convergence of Marshaled Learning (at around 200 global rounds) is much sooner than that of FedAvg (at around 400 global rounds) since dynamic knowledge propagation in Marshaled Learning can effectively harness data heterogeneity and accelerate the training.

C.2. The Selection of Dynamic Point

We investigate the dynamic point selection in Marshaled Learning, which initiates dynamic knowledge propagation, using ResNet-34 and the Cifar-100 dataset. ResNet-34 consists of 4 residual blocks, each with 2 residual layers. We define four dynamic point cases, each positioned after a different residual block: after residual block 1, 2, 3, and 4. For clarity, we label these dynamic points as d_1, d_2, d_3, and d_4. To isolate the impact of client and cluster configurations, we test with 2 and 3 clusters in Marshaled Learning, with each cluster containing 5 clients (10 and 15 clients total). The results are summarized in Table 6.

The results indicate that the dynamic point selection has only a minimal effect on performance. Specifically, we observed that the top-1 accuracy of Marshaled Learning varied by only about 1.2% across all possible dynamic point positions, regardless of whether there were 10 or 15 clients in Marshaled Learning. These findings suggest that the choice of dynamic point location has a negligible impact on model performance. As a result, we conclude that the dynamic point should be placed in a layer that minimizes communication costs in Marshaled Learning.

Algorithm 1: Marshaled Learning

```
1: Input: neural network $f$, number of clients $M$, number of subnets $N$, maximum client memory $Mem$, number of global
   communication rounds $T$, number of local epochs $E$, dynamic point $d$.
2: Server Initialization: 1) divide $f$ into $N$ subnets: $f = f_N \circ f_{N-1} \circ \cdots \circ f_1$, and each subnet $f_i$ can fit into the client's
   TEE memory $Mem$; 2) group clients into $K = \lfloor M/N \rfloor$ clusters, each consisting of $N$ clients.
3: for each global round $t_g \in \{1, \cdots, T_g\}$ do
4:   for all clusters $k \in \{1, \cdots, K\}$ in parallel do
5:     for each client $i \in \{1, \cdots, N\}$ in cluster k do
6:       // Train the neural network on data located at client $i$
7:       Assign the first subnet to client $i$ and encrypt it by TEE.
8:       Send encrypted subnet into TEE for client $i$ and store it inside TEE.
9:       Set up dynamic propagation paths $\mathcal{P}$ based on dynamic point $d$ (Section 3.3).
10:      // Perform dynamic knowledge propagation.
11:      for $Path \in \mathcal{P}$ do
12:        //Mitigate straggler issues.
13:        if client $j \in Path$ is a straggler then
14:          Replace client $j$ with the normal client from another cluster.
15:        end if
16:        Perform forward and backward propagation on clients following $Path$ for $E$ local epochs (Section 3.2).
17:        //During the forward and backward process
18:        TEE of client $i$ decrypts the subnet and processes it inside.
19:        TEE of client $i$ encrypted the intermediate results and sent it to client $i + 1$ during the forward propagation.
20:        TEE of client $i$ encrypted the gradients and sent it to client $i - 1$ during the backward propagation.
21:        Average the parameters of each subnet before dynamic point $f_i, i \in [1, d)$ across clusters and update $f_i$. In this
           process, clients' TEEs encrypt and exchange the subnet parameters with other TEEs.
22:      end for
23:    end for
24:  end for
25:  Aggregate the parameters of each subnet across clusters and update $f$. TEE encrypted the exchange parameters and
     communicated with other TEEs to achieve the aggregation.
26: end for
27: Output: neural network $f$.
```

Table 6. The dynamic point selection in Marshaled Learning is evaluated on the CIFAR-100 dataset under a non-iid setting. Four dynamic point positions are examined, each placed after a different residual block in the ResNet-34 architecture.

Number of Clients	Dynamic Point d	Top-1 Accuracy
10	d_1	62.3%
10	d_2	62.7%
10	d_3	63.5%
10	d_4	62.5%
15	d_1	63.0%
15	d_2	63.8%
15	d_3	62.6%
15	d_4	63.6%

Table 7. Performance comparison across different non-iid degrees. We evaluate Marshaled Learning and FedAvg on the Cifar-100 dataset, using ResNet-34 for Marshaled Learning and ResNet-9 for FedAvg.

Non-iid Degree β	Marshaled Learning	FedAvg
0.1	55.0%	54.3%
0.2	61.0%	57.3%
0.3	62.6%	58.5%
0.4	63.6%	58.8%
0.5	63.8%	58.8%
0.6	65.0%	60.4%

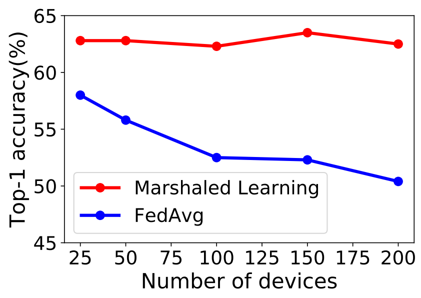

Figure 4. Impact of client numbers on ResNet networks with the CIFAR-100 dataset under non-iid settings.

C.3. The Impacts of Non-iid Degrees

We investigate the impact of non-iid degrees on the performance of Marshaled Learning. In this experiment, we evaluate Marshaled Learning using different non-iid partitions of the Cifar-100 dataset, with values of β set to $0.1, 0.2, 0.3, 0.4, 0.5$, and 0.6. A smaller β corresponds to a higher non-iid degree. We use FedAvg, a popular federated learning framework, to compare the performance. In Marshaled Learning, we deploy a ResNet-34 network, while in FedAvg, we use ResNet-9. The number of clients in both Marshaled Learning and FedAvg is set to 15, with 3 clusters in Marshaled Learning, each containing 5 clients. The results are presented in Table 7.

We find that Marshaled Learning consistently outperforms FedAvg across all non-iid degrees. As β decreases to 0.1, the performance of Marshaled Learning slightly drops, but it still surpasses FedAvg. However, as β increases, Marshaled Learning shows a significant performance boost, gaining an advantage of around 4% to 5% over FedAvg. These results highlight the effectiveness of Marshaled Learning in handling varying non-iid conditions.

C.4. Accuracy Impact for a Large Number of Clients.

We explore the scalability of Marshaled Learning with an increasing number of clients, which allows us to assess the effectiveness and robustness of Marshaled Learning under more realistic and challenging scenarios involving a larger distributed client. In the previous setting, Marshaled Learning was evaluated with up to 25 clients. In this experiment, we increase the number of clients to 50, 100, and 200 to demonstrate its scalability. We assume that the data held by each client does not overlap, leading to a decrease in the number of samples per client as the client count increases. This exacerbates the challenges associated with data heterogeneity. The comparison uses the ResNet architecture and the non-iid Cifar-100 dataset, with Marshaled Learning and FedAvg as the methods under evaluation.

Figure 4 illustrates the scalability of Marshaled Learning under a large number of clients. The increased number of clients from 25 to 200 does not reduce the prediction performance of Marshaled Learning. However, in contrast, as the number of clients increased, the performance of FedAvg gradually decreased from 58% to 50%. Therefore, our findings confirm that Marshaled Learning outperforms federated learning when dealing with a large number of clients.

Table 8. Model performance comparison under heterogeneous dataset sizes using the Cifar-100 testing dataset. Among 15 clients, 5 are assumed to have larger training datasets, each containing twice the number of samples as the other clients.

Networks	Metrics	Marshaled Learning	FedAvg
ResNet	Top-1	**59.52%**	54.15%
	Top-5	**84.30%**	82.60%
	Top-10	**89.87%**	88.62%
EfficientNet	Top-1	43.55%	**45.21%**
	Top-5	**73.91%**	71.11%
	Top-10	**83.84%**	79.02%
ViT	Top-1	**90.00%**	84.42%
	Top-5	**98.72%**	97.32%
	Top-10	**99.40%**	98.80%

Table 9. Top-1 accuracy comparison of VIT models on TinyImageNet.

Clients	Marshaled Learning	FedAvg	FedProx	SplitFL	PPFL
12	**84.56%**	81.12%	81.26%	84.42%	26.63%

C.5. Experiments of Heterogeneous Data

In this experiment, we study a heterogeneous data scenario where the size of the client's data varies: five out of 15 clients hold twice as many samples as the remaining ten. These 15 clients are used across three network backbones (ResNet, EfficientNet, and ViT), with the larger-data clients grouped into one cluster. We evaluate on the CIFAR-100 dataset and compare the training performance of Marshaled Learning against the FedAvg baseline, reporting top-1, top-5, and top-10 accuracy for comprehensive analysis.

As shown in Table 8, Marshaled Learning consistently outperforms FedAvg across all settings, with one exception: FedAvg achieves slightly higher top-1 accuracy on EfficientNet. However, even in this case, Marshaled Learning yields better top-5 and top-10 accuracy, suggesting that it provides more robust and reliable performance across a broader range of predictions. These results highlight Marshaled Learning 's overall advantage, particularly in scenarios with varying client data distributions.

We believe the slight outperformance of FedAvg may be due to EfficientNet being a lightweight network, which can result in poorer top-1 accuracy when data samples vary across different clients. Since Marshaled Learning can deploy a larger EfficientNet model (EfficientNet-b4) compared to FedAvg (EfficientNet-b0), the top-5 and top-10 metrics highlight the performance benefits of larger-scale networks, demonstrating the advantage of Marshaled Learning.

Therefore, based on the training performance across three networks, we demonstrate that Marshaled Learning can outperform traditional FL in scenarios with variable client dataset sizes.

C.6. Experiments Results on TinyImageNet

To further validate our results on CIFAR-10 and CIFAR-100, we conducted experiments on the more complex TinyImageNet dataset using the ViT model. The model and participant settings follow Section 4.1, and the results are shown in Table 9. These results support the same conclusion as in Section 4.2: Marshaled Learning leverages large-scale neural networks more effectively than the baseline methods.

C.7. Scalability Experiments on TEE

We varied the cluster size to further investigate the computational overhead of TEEs and assess the scalability of our framework. Specifically, we increased the number of participating clients per cluster from two to three and four. To avoid the gRPC server's maximum byte limitation, we set the batch size to 100 for all network structures. The models used (ResNet, EfficientNet, and ViT) are the same as in the previous TEE experiments (Section 4.4). We test these results on the same type but a different Virtual Machine on Azure. The corresponding results are presented in Table 10, 11, and 12. From these results, we observe that increasing the number of clients per cluster leads to a slight increase in the overall computational cost for both the Non-TEE and TEE settings. This additional cost primarily stems from the increased communication overhead, which is

Table 10. The computation comparison between non-TEE and TEE environment, two clients per cluster.

Networks	Environment	Execution time (s/batch)	Overhead
ResNet	Non-TEE	2.97	1×
	TEE	5.73	1.93×
EfficientNet	Non-TEE	0.29	1×
	TEE	1.76	6.06×
ViT	Non-TEE	4.29	1×
	TEE	10.22	2.38×

Table 11. The computation comparison between non-TEE and TEE environment, three clients per cluster.

Networks	Environment	Execution time (s/batch)	Overhead
ResNet	Non-TEE	2.93	1×
	TEE	6.12	2.08×
EfficientNet	Non-TEE	0.33	1×
	TEE	2.02	6.12×
ViT	Non-TEE	4.35	1×
	TEE	11.53	2.65×

Table 12. The computation comparison between non-TEE and TEE environment, four clients per cluster.

Networks	Environment	Execution time (s/batch)	Overhead
ResNet	Non-TEE	3.01	1×
	TEE	6.44	2.13×
EfficientNet	Non-TEE	0.35	1×
	TEE	2.16	6.17×
ViT	Non-TEE	4.42	1×
	TEE	12.75	2.88×

Table 13. The partition strategy for two clients per cluster in TEE experiments.

Models	Part 1	Part 2
ResNet	init layer + 10 residual blocks	6 residual blocks + avgPool + fc layers
EfficientNet	init layer + 9 mobile blocks	3 mobile + 1 conv block + avgPool + fc layers
ViT	patch embedding + 6 transformer blocks	6 transformer blocks + fc layers

Table 14. The partition strategy for three clients per cluster in TEE experiments.

Models	Part 1	Part 2	Part 3
ResNet	init layer + 7 residual blocks	5 residual blocks	3 residual blocks + avgPool + fc layers
EfficientNet	init layer + 8 mobile blocks	4 mobile blocks	1 conv block + avgPool + fc layers
ViT	patch embedding + 4 transformer blocks	4 transformer blocks	4 transformer blocks + fc layers

more expensive to handle under the TEE setting. However, the TEE overhead does not grow significantly and remains within an acceptable range, especially for larger-scale networks, such as ResNet and ViT. Overall, these results demonstrate that our framework maintains good scalability as the number of clients within a cluster increases.

The model separation strategies for all the TEE settings (two, three, and four clients per cluster) are shown in Tables 13, 14, and 15. Based on the results in Tables 10, 11, and 12, we conclude that the separation has minimal impact on computational cost, since the Non-TEE setting execution time varies minimally under the increased communication time.

Table 15. The partition strategy for four clients per cluster in TEE experiments.

Models	Part 1	Part 2	Part 3	Part 4
ResNet	init layer + 6 residual blocks	5 residual blocks	4 residual blocks	3 residual blocks + avgPool + fc layers
EfficientNet	init layer + 6 mobile blocks	4 mobile blocks	2 mobile blocks	1 conv block + avgPool + fc layers
ViT	patch embedding + 3 transformer blocks	3 transformer blocks	3 transformer blocks	3 transformer blocks + fc layers

Table 16. The comparison between Marshaled Learning with/without straggler mitigation on the non-iid CIFAR-100 dataset. We designate the last client in the second cluster as the straggler and set the dynamic point $d = 2$, where dynamic propagation begins after client 2.

Straggler degree	Method	Speed Up	Accuracy
Light	Marshaled Learning without Straggler Mitigation	1×	62.7%
	Marshaled Learning	1.33×	58.1%
Severe	Marshaled Learning without Straggler Mitigation	1×	62.7%
	Marshaled Learning	2.78×	57.0%

Table 17. The comparison between Marshaled Learning with/without straggler mitigation on the non-iid CIFAR-100 dataset. We designate the second and third clients in the second cluster as stragglers and set the dynamic point $d = 3$, where dynamic propagation begins after client 3.

Straggler degree	Method	Speed Up	Accuracy
Light	Marshaled Learning without Straggler Mitigation	1×	63.5%
	Marshaled Learning	1.33×	58.2%
Severe	Marshaled Learning without Straggler Mitigation	1×	63.5%
	Marshaled Learning	2.78×	46.9%

C.8. Straggler Poisition Analysis:

In this subsection, we further analyze the impact of the position and numbers of the straggler in Marshaled Learning. In the previous setting, the straggler was the second client of the second cluster, before the dynamic point $d = 2$. Here, we move the straggler to the last client of the second cluster, placing it after the dynamic point. We refer to this as the deeper straggler case. These results show that a straggler handling deeper layers of the model experiences a greater accuracy drop ($4\% \sim 6\%$) than one handling earlier layers($3\% \sim 5\%$), compared with the results of Table 4. This finding suggests that the position of a straggler within the model has impacts on overall performance, and that stragglers in deeper layers require additional mitigation (*e.g.* move the straggler to an earlier position) to reduce accuracy loss.

We also investigate a *multi-straggler case*, where the second and third clients of the second cluster before the dynamic point $d = 3$ (start dynamic propagation after the third client) are both stragglers. The comparison results of our straggler mitigation has been shown in Table 17. Comparing Table 17 and Table 4, the multi-straggler case shows a 2.5% greater accuracy drop under the "Light" straggler degree and a 12% greater drop under the "Severe" degree compared to our previous setting in the main paper. These results show that as the number of stragglers increases, model updates become more imbalanced, causing greater accuracy loss and indicating a positive correlation between accuracy loss and update imbalance.

D. Details of TEE settings

TEE Devices: We apply Intel SGX on Azure VM as our TEE. We choose type DC8s-v3 as our VM. More details can be found in the Azure documentation: https://learn.microsoft.com/en-us/azure/confidential-computing/confidential-vm-faq.

Dataset: We only use Cifar-10 dataset in TEE experiments. We do not consider the heterogeneity in TEE experiments.

Network Architectures: We select the same network architectures in non-TEE experiments. We remove the batch-normalized layers in these networks to avoid the gradient disappearing in our experiment.

Libraries and Tools: We use Tensorflow v1 to implement Marshaled Learning on dockers of Linux. In our implementation, we apply the lightweight library Gramine, which helps applications run in the SGX enclave. Gramine provides a signing key for Intel SGX, which the enclave will only load the signed applications. The signing key is generated using the RSA 3072 algorithm.

To achieve transfer data between clients in Marshaled Learning, we implement the gRPC, an open-source connection service, to send the forward data and gradients. To encrypt the channel of gRPC, we use TLS protocols, which apply the

RSA 2048 algorithm to generate the encryption key and certificates to secure the transportation channel. To accommodate the TLS protocols into SGX, we use RA-TLS protocols, which establish TLS connections in the TEE. The RA (remote attestation) also provides the X.509 certificate to verify the integrity of the transmission process. In practice, we first generate private keys (.pem) and certificates (.ca) for the channels and deploy them to the TEEs. When a communication channel is established, the server verifies the certificate and uses the public key to encrypt outgoing data, while the private key is used to decrypt incoming data.

Marshaled Learning Implementation: We simplify the Marshaled Learning setup for the TEE experiments. In this configuration, we assume two clusters, each containing two, three, or four clients. Each cluster deploys a single network, partitioned so that each client hosts a portion of the model. During each training iteration, the first client in a cluster performs its part of the forward pass and sends the intermediate output to the second client via a secure gRPC connection; this process continues sequentially until the final client. The last client completes the forward pass, computes the gradients, and sends them back to the preceding clients using secure gRPC, allowing each client in reverse order to update its portion of the network. After all clusters complete one training epoch, the TensorFlow training server collects the gradients from the clients and returns the aggregated results, with the aggregation also transmitted over secure gRPC. We use the federated learning code from the Confidential Computing Zoo (CCZoo) for our implementation. In the TEE experiments, we use two types of servers to aggregate model parameters and handle client communication: a model aggregation server and multiple communication servers. The aggregation server is provided by the open-source CCZoo framework, while the communication servers are gRPC servers running on different ports. We use Google Protocol Buffers (protobuf) to define the data structures and service interfaces for these gRPC servers.

Non-TEE and TEE environment: In this experiment, the non-TEE environment means using CPUs on Azure VM to train the Marshaled Learning outside SGX enclave, and the TEE environment means using the same CPUs on Azure VM to train the Marshaled Learning inside SGX enclave.

E. Future Works

In this paper, we consider the feasibility of accommodating large-scale models into limited resource constrained TEEs (Investigate the Intel SGX). With the development of TEE techniques, in the future we will consider more types of TEEs (*e.g.*, Intel TDX and ARM TrustZone). We do not consider the communication overhead in this work, since it is an open question in the collaborative learning approaches that support large-scale models on edge devices. We will consider this challenge in the future.