Received 22 November 2024, accepted 5 January 2025, date of publication 10 January 2025, date of current version 15 January 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3528248

APPLIED RESEARCH

Strategy-Switch: From All-Reduce to Parameter Server for Faster Efficient Training

NIKODIMOS PROVATAS[1], IASONAS CHALAS[1], IOANNIS KONSTANTINOU[2], AND NECTARIOS KOZIRIS[1], (Member, IEEE)
[1]School of Electrical and Computer Engineering, National Technical University of Athens, 11527 Athens, Greece
[2]Department of Informatics and Telecommunications, University of Thessaly, 35100 Lamia, Greece

Corresponding author: Nikodimos Provatas (nprov@cslab.ece.ntua.gr)

This work was supported by European Regional Development Fund of the European Union and Greek National Funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the Call RESEARCH—CREATE—INNOVATE under Project T2EDK-01231.

ABSTRACT Deep learning plays a pivotal role in numerous big data applications by enhancing the accuracy of models. However, the abundance of available data presents a challenge when training neural networks on a single node. Consequently, various distributed training methods have emerged. Among these, two prevalent approaches are *All-Reduce* and *Parameter Server*. *All-Reduce*, operating synchronously, faces synchronization-related bottlenecks, while the *Parameter Server*, often used asynchronously, can potentially compromise the model's performance. To harness the strengths of both setups, we introduce *Strategy-Switch*, a hybrid approach that offers the best of both worlds, combining speed with efficiency and high-quality results. This method initiates training under the *All-Reduce* system and, guided by an empirical rule, transitions to asynchronous *Parameter Server* training once the model stabilizes. Our experimental analysis demonstrates that we can achieve comparable accuracy to *All-Reduce* training but with significantly accelerated training.

INDEX TERMS Deep learning, distributed systems, all-reduce, parameter server.

Recent years have witnessed an exponential interest increase in the field of deep learning [1]. Such techniques have been widely adopted across various scientific domains, particularly in applications such as image classification [2], [3], [4], segmentation [5], [6], [7], speech recognition [8], [9], [10], and text classification [11], [12], [13]. Adaptation and evolution of network architectures to improve predictive precision remains an ongoing pursuit. Derived from the image classification domain, a notable example is identified on the Imagenet dataset [14], where continuous efforts persist to enhance its prediction accuracy [15], [16], [17], [18], [19], [20].

Concurrently, the volume in available data [21], [22] has rendered training on a single computing node time-prohibitive, even when leveraging accelerators like GPUs. Consequently, to tackle the explosion in data amount, engineers and researchers have turned to distributed environments employing data parallelism. Such distributed setups, adopted widely across various distributed execution systems [23], [24], [25], [26], involve participating machines utilizing different segments of training data and local model copies to generate a unified global model.

To address the need for distributed learning, researchers have proposed varied architectures based on data parallelism. Notably, two fundamental distributed deep learning architectures, *All-Reduce* [27], [28], [29] and *Parameter Server* [30], [31], [32], [33], have emerged. *All-Reduce* is a decentralized architecture where participating peers synchronize computed gradients using all-reduce primitives, updating their local models. Conversely, *Parameter Server* is a centralized architecture featuring servers hosting key-value stores for the global model, with workers training local model copies and updating the global model via gradients sent to the server. Workers under the *Parameter Server* can either perform the server model update synchronously

The associate editor coordinating the review of this manuscript and approving it for publication was Fabrizio Messina.

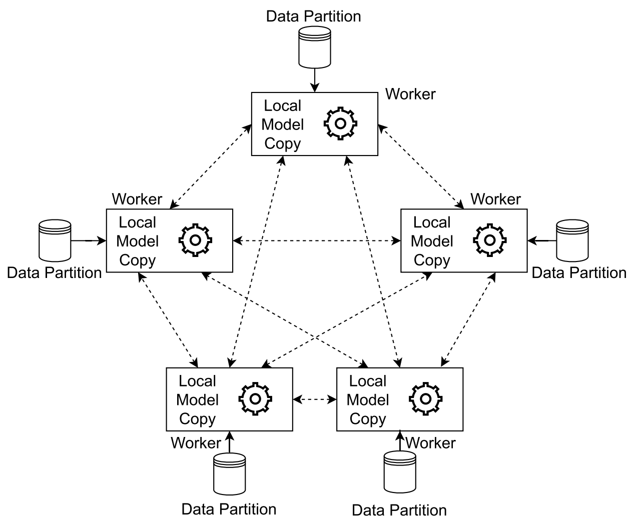

FIGURE 1. ***All-Reduce*** **architecture: Dashed lines indicate communication, while solid lines indicate data extraction.**

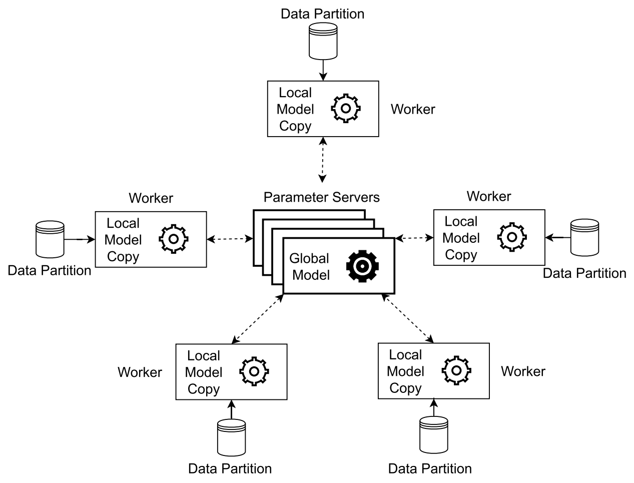

FIGURE 2. ***Parameter Server*** **architecture: Dashed lines depict communication, solid lines represent data extraction.**

or asynchronously, but lack of synchronization is usually preferred when adopting *Parameter Server*.

While *All-Reduce* and *Parameter Server* offer distinct advantages and drawbacks, their nature yields contrasting challenges. Synchronous decentralized architectures like *All-Reduce* face synchronization-related overheads [34] but exhibit convergence akin to single-node training. In contrast, the centralized nature of *Parameter Server* risks network hotspots in the server [35], potentially compromising model quality due to outdated parameters [36]. Despite synchronization hurdles, asynchronous training in *Parameter Server* expedites the training process.

In light of these considerations, there is a growing need to clarify the interplay between deep learning convergence requirements and system-level performance constraints, in order to accommodate practitioners from both the distributed computing and machine learning domains. In this work, inspired by *Sync-Switch* [37], and acknowledging these architectural differences, we propose *Strategy-Switch*, a method that unifies *All-Reduce* and asynchronous *Parameter Server*, attempting to ensure training stability alongside fast execution. Our main contributions include:

1) Presenting a benchmarking analysis outlining disparities between *All-Reduce* and asynchronous *Parameter Server*.
2) Introducing *Strategy-Switch* as a hybrid training method, evaluating its performance using established image classification benchmarks.
3) Proposing an empirical rule governing the transition from *All-Reduce* to *Parameter Server*.

The remainder of this paper is structured as follows: Section I provides background information on distributed deep learning architectures, followed by a comparative benchmarking analysis in Section II. Section III details *Strategy-Switch*, evaluated in Section IV, which also delves into the proposed empirical transition rule. Section V discusses related work, highlighting the distinctions between *Strategy-Switch* and *Sync-Switch*, while Section VI concludes our work.

I. THEORETICAL BACKGROUND

In this section, we establish the necessary background on the two distributed training paradigms integrated into the *Strategy-Switch* training methodology.

A. ALL-REDUCE

All-Reduce [27], [28], [29] embodies a decentralized training methodology where all workers engage in gradient exchange. This process is illustrated in Fig. 1. At the onset of a training step, each worker extracts a mini-batch from their local data partition, where they compute gradient vectors. Synchronous all-reduce techniques facilitate the exchange and aggregation of these gradients among workers and subsequently update each worker's local model for the next iteration. This synchronous nature ensures uniformity across local models when the next iteration is initiated. A widely adopted *All-Reduce* approach within various learning systems like TensorFlow [26] is the Ring All-Reduce [38] method.

B. PARAMETER SERVER

Parameter Server [30], [31], [32], [33] constitutes a centralized approach for distributed model training. In this setup, servers maintain a global model, whereas workers compute gradients on local model copies, using data from a data partition assigned to them, as depicted in Fig. 2. *Parameter Server* training can operate either in bulk synchronous (BSP) or asynchronous parallel (ASP) mode. During a training step, workers retrieve the latest model parameters from servers, compute gradients using their local data, and push these gradients back to update the global model on the servers. In ASP, servers update the global model upon receiving new

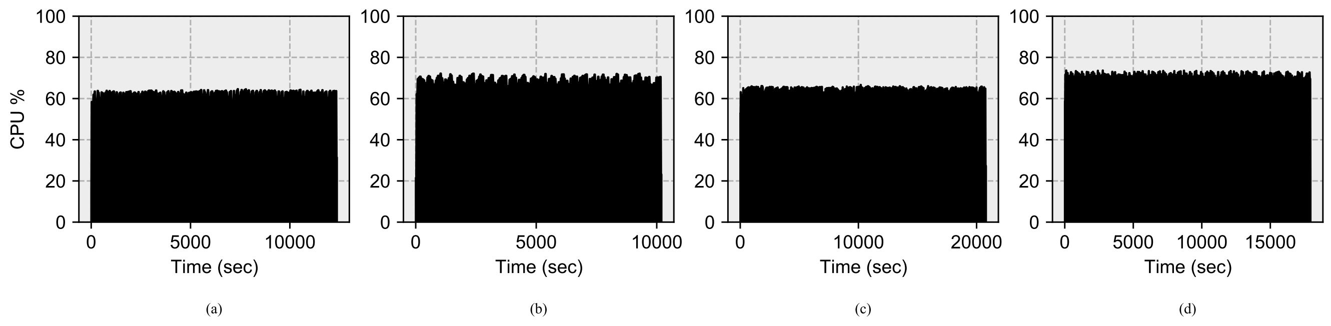

FIGURE 3. #C1 cluster CPU utilization for training benchmarks under *All-Reduce* and *Parameter Server* (a) #B1 -*All-Reduce* (b) #B1 -*Parameter Server* (c) #B2 -*All-Reduce* (d) #B2 -*Parameter Server*.

TABLE 1. Benchmark overview.

Benchmark	Dataset	Network	Hyperparameters
#B1	CIFAR-10	ResNet-20v1 with 0.27M parameters	SGD, mini-batch size 128, momentum 0.9, step decay
#B2	CIFAR-100	ResNet-32v1 with 0.48M parameters	Learning rate (initial 0.1), divided by 0.1 at epochs 82 and 133

gradients, whereas in BSP, they wait for gradients from all workers. Throughout this paper, any reference to *Parameter Server* without specifying the training mode refers to ASP. Although ASP is faster since it lacks synchronization-related performance overheads, it may suffer from stale gradients, potentially compromising model accuracy.

C. HYPERPARAMETERS IN DISTRIBUTED SETTINGS

In these training paradigms, hyperparameters such as mini-batch size and learning rate are adjusted to emulate equivalent single-node setups [39]. To mirror single-node training, the global batch size should match the single-node counterpart, resulting in a scaled per-worker size based on the worker count. In asynchronous *Parameter Server*, the learning rate is also adjusted since each worker contributes independently to the global model located on the servers.

II. BENCHMARKING ALL-REDUCE AND ASYNCHRONOUS PARAMETER SERVER

This section presents an experimental evaluation of training using the *All-Reduce* and *Parameter Server* paradigms to delve deeper into their characteristics.

A. EXPERIMENTAL SETUP

The evaluation involved two benchmarks utilizing CIFAR datasets [40] and ResNet-based neural networks [41] as outlined in Table 1.

CIFAR-10 [40] is a set of labeled images. The CIFAR-10 dataset consists of 60000 32×32 color images divided into 10 classes, with 6000 images per class. The dataset is divided into 50000 training data and 10000 test data. The dataset is divided into five training batches and one control

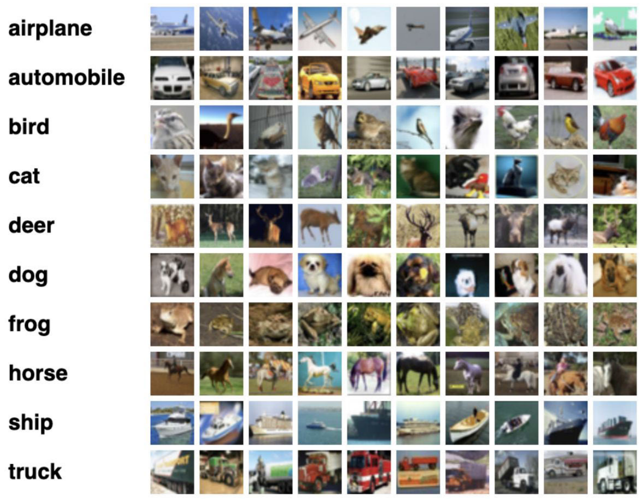

FIGURE 4. Examples of CIFAR-10 images for each category.

batch, each of which consists of 10000 images. The control batch contains exactly 1000 randomly selected images from each class. Training batches contain the remaining images in random order, but some training batches may contain more images from the same class. In total the training batches contain exactly 5000 images of each class. The classes into which the data is divided are: plane, car, bird, cat, deer, dog, frog, horse, ship, and truck. In Fig. 4 we see examples of images from each class.[1] The CIFAR-100 dataset is similar to CIFAR-10, but it contains 100 classes with 600 images each. For each class, there are 500 training images and 100 testing images. The 100 classes in CIFAR-100 are organized into 20 superclasses. Each image has both a "fine" label, indicating its specific class, and a "coarse" label, representing its superclass.

We adopt the hyperparameters referenced in Table 1 adjusted for distributed setups, as explained in Section I-C. We utilized a private Openstack cloud cluster that provided us with virtual machines to setup our evaluation infrastructure. Our infrastructure consists of a homogeneous

[1]downloaded from https://www.cs.toronto.edu/ kriz/cifar.html

TABLE 2. VM specifications in clusters.

VM Flavor	vCPUs	RAM	HDD Storage
m1.xxlarge1	8	32GB	100GB
m1.large3	4	16GB	60GB
regale-small	2	8GB	30GB

TABLE 3. Clusters used for experiments.

Cluster Code	Cluster Type	Number of VMs per Flavor		
		m1.xxlarge1	m1.large3	regale-small
#C1	Homogeneous	5	-	-
#C2	Heterogeneous	5	2	2

cluster (i.e., the cluster nodes hardware configuration is the same) and a heterogeneous cluster (i.e., the cluster nodes hardware configuration varies between the nodes). Table 2 details the specifications of the Openstack cluster virtual machine flavors utilized in the clusters. In Table 3 we provide an overview of the clusters utilized in our experiments. We employed TensorFlow v2.6.2 for both *All-Reduce* and asynchronous *Parameter Server* training in each cluster.

B. HOMOGENEOUS #C1 CLUSTER BENCHMARKING

Fig. 3 illustrates the CPU utilization of *All-Reduce* and *Parameter Server* training in the homogeneous #C1 cluster. For both benchmarks, #B1 and #B2, *All-Reduce* and *Parameter Server* display similar CPU utilization in this environment. *All-Reduce* hovers around 65%, while *Parameter Server* peaks at approximately 75%. The *All-Reduce* strategy exhibits lower performance due to synchronization overheads, whereas *Parameter Server* demonstrates improved cluster utilization as workers update the model independently.

The network traffic in #C1 (Fig. 5) confirms similar levels of incoming and outgoing traffic for *All-Reduce* across both benchmarks, approximately 50 MB/sec. In contrast, *Parameter Server* manifests higher network usage with approximately 65 MB/sec for #B1 and 70 MB/sec for #B2. This disparity results from the increased complexity of the ResNet-32 model used in #B2, leading to a greater exchange of gradients and model parameters.

C. HETEROGENEOUS #C2 CLUSTER BENCHMARKING

Fig. 6 displays the CPU utilization of *All-Reduce* and *Parameter Server* training in the heterogeneous #C2 cluster. Compared to the homogeneous cluster, *All-Reduce* and *Parameter Server* showcase significantly divergent CPU utilization. *All-Reduce* demonstrates lower CPU usage at $\sim$50% for both benchmarks, while *Parameter Server* exhibits much higher usage ($\sim$80%). The synchronous nature of *All-Reduce* results in varied worker speeds, impacting performance significantly in the heterogeneous environment. This impact is due to the different computing capabilities of the participating nodes and the requirement for synchronization. The synchronization requirement stalls faster nodes that need to wait for the slower nodes computation to finish. Conversely, *Parameter Server*, operating asynchronously, presents consistent utilization irrespective of worker differences, making it more adaptable to cluster heterogeneity. This utilization consistency is the result of the asynchronous computation: faster nodes do not need to wait for slower nodes to finish their computations before gradient updates. Nevertheless, this speed comes at the expense of slower convergence, since "stale" computations of slower nodes diverge the gradient computation away from its correct value by adding noise at every computation step.

In terms of network traffic (Fig. 7), *All-Reduce* exhibits similar levels compared to the homogeneous cluster, while *Parameter Server* doubles the network traffic on #C2 compared to #C1. The higher traffic validates the effect of stale value updates of slower nodes, as more message exchanges (therefore more network traffic) between nodes are needed in order to come to the same conclusion with the homogeneous setup.

III. STRATEGY-SWITCH

In this section, we present our approach on distributed training, *i.e.* the *Strategy-Switch*. We provide an extensive discussion on the design of *Strategy-Switch* and conclude with an empirical rule on the switching point.

A. APPROACH

In order to combine both the benefits of the two distributed training setups discussed in Section I, we propose *Strategy-Switch*. *Strategy-Switch* is a hybrid distributed setup, which performs the model training in an *All-Reduce* approach up to a specific epoch and then proceeds with training under an asynchronous *Parameter Server* setup. *Strategy-Switch-*α*%* is illustrated in Algorithm 1, where α% represents the percentage of epochs performing *All-Reduce* training.

After the completion of the last *All-Reduce* epoch, *Strategy-Switch* dumps (saves) the global model on a Distributed File System (DFS), so that the parameter servers (PS) can load it as the initialization point for the subsequent asynchronous training stage. In practice, this model exchange procedure requires minimal overhead, as the corresponding checkpoint is copied to the DFS and then broadcast by the servers.

B. THEORETICAL EXPLANATION

1) CONVERGENCE IN STRATEGY-SWITCH

In *Sync-Switch* [37], the authors provide a detailed theoretical explanation on why changing training in the *Parameter Server* architecture from BSP mode to ASP could benefit the training process (Section IV-A). In our proposed *Strategy-Switch*, the BSP *Parameter Server* phase is replaced with an *All-Reduce* phase. Therefore, in *Strategy-Switch*, we can consider *All-Reduce* training as a stable approach to lead the model parameters closer to the optimization point, simillar to the approach of the *Sync-Switch* [37]. When gradients are small, smaller movements to the model parameters are

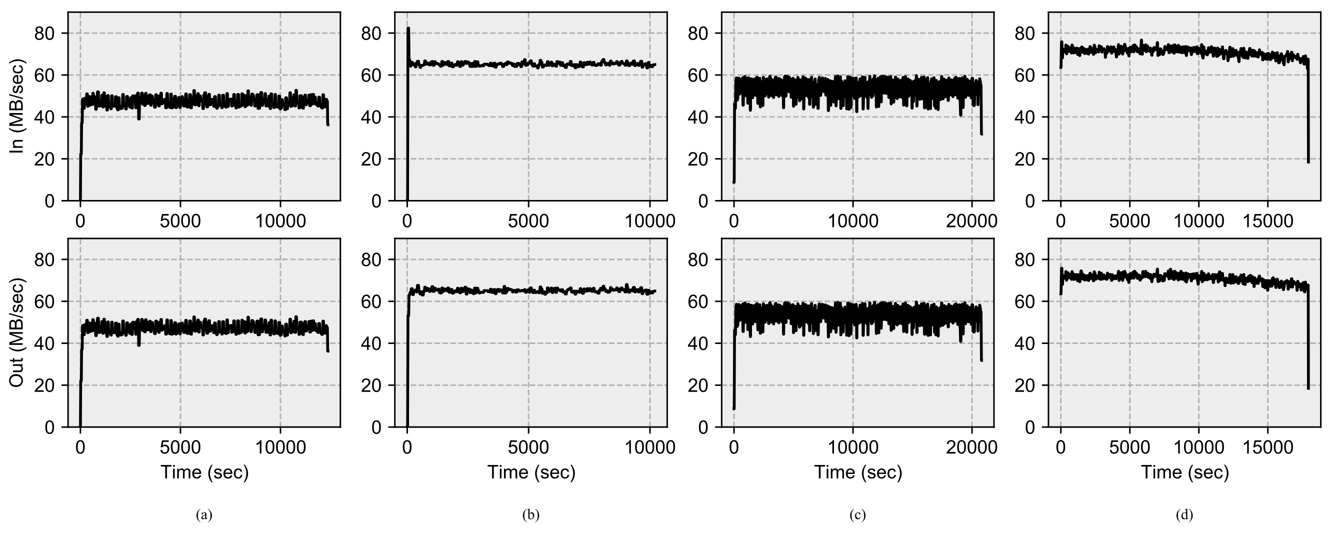

FIGURE 5. #C1 cluster network utilization for training the benchmarks under *All-Reduce* and *Parameter Server* (a) #B1 -*All-Reduce* (b) #B1 -*Parameter Server* (c) #B2 -*All-Reduce* (d) #B2 -*Parameter Server*.

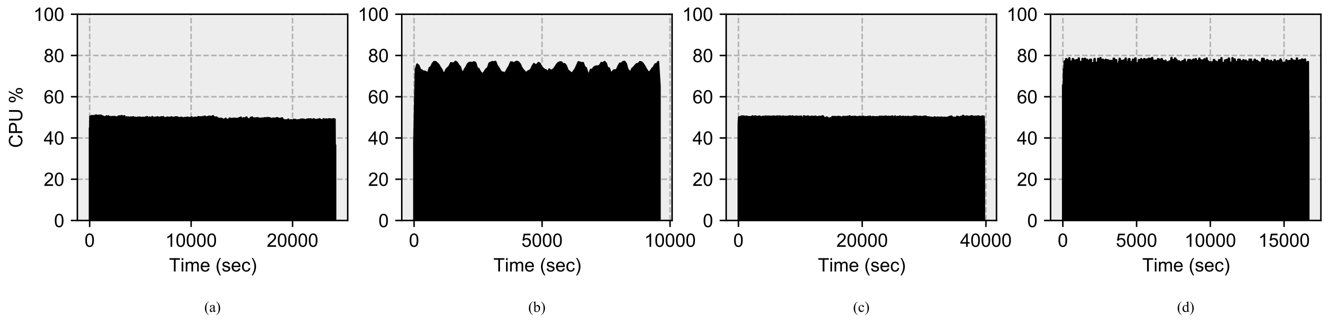

FIGURE 6. #C2 cluster CPU utilization for training the benchmarks under *All-Reduce* and *Parameter Server* (a) #B1 -*All-Reduce* (b) #B1 -*Parameter Server* (c) #B2 -*All-Reduce* (d) #B2 -*Parameter Server*.

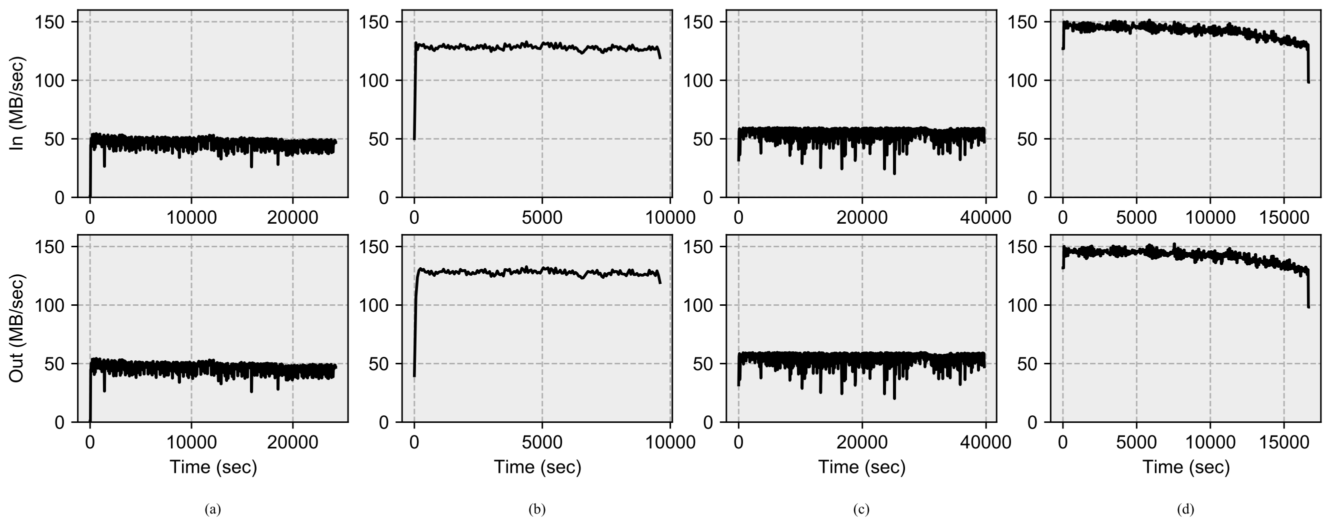

FIGURE 7. #C2 cluster network utilization for training the benchmarks under *All-Reduce* and *Parameter Server* (a) #B1 -*All-Reduce* (b) #B1 -*Parameter Server* (c) #B2 -*All-Reduce* (d) #B2 -*Parameter Server*.

caused. Hence, we can consider that the model is not crucially altered and the small change of the model parameters render it less vulnerable to stale gradients that may occur in asynchronous learning setups.

Algorithm 1 *Strategy-Switch-α%*

```
Input: Trainable model $m$, Training and validation data $d$, Global hyperparameters hp, Number of training epochs $e$, Percentage of epochs under All-Reduce $\alpha$
Output: Trained model $m$
 1: Deploy a set w_allred of All-Reduce workers
 2: for $i = 1$ to $\alpha * e$ do
 3:     train(w_allred, m, d, hp, i)
 4: end for
 5: Dump m on DFS
 6: Initiate a set ps of parameter servers
 7: ps.load(m)
 8: Deploy a set w_ps of Parameter Server workers
 9: for $i = \alpha * e + 1$ to $e$ do
10:     train(w_ps, ps, m, d, hp, i)
11: end for
12: return $m$
```

2) WHY ALL-REDUCE OVER BSP PARAMETER SERVER

As shown in Section II, asynchronous *Parameter Server* consumes more network compared to *All-Reduce*. However, the higher CPU utilization attributed to the lack of synchronization leads to faster training. When using BSP *Parameter Server*, the synchronization will impact cluster utilization and the training will also be prone to possible network bottlenecks. Such bottlenecks may be attributed to either server hotspots [35] and network waiting time for synchronization [42]. Therefore, for synchronous training, it is more efficient to exploit *All-Reduce* instead of BSP *Parameter Server*, since network hotspots will be avoided, due to *All-Reduce* being decentralized. Various studies [43], [44], [45] discussing the network efficiency of *All-Reduce* compared to *Parameter Server* support our claim.

C. EMPIRICAL RULE FOR THE SWITCHING POINT

The design of *Strategy-Switch* raises one important question: When is it the right time to change from *All-Reduce* to *Parameter Server* training? In this section, we discuss an empirical rule which decides online throughout the training process whether it is the right time to proceed with *Parameter Server* training. Suppose $v_{loss}(i)$ is the validation loss in epoch i and k is the last training epoch finished, then when the boolean expression in 1 becomes true, the training continuous under an asynchronous *Parameter Server* setup.

$$s = \frac{\sum_{i=0}^{4} \|v_{loss}(k-i) - v_{loss}(k-i-1)\| \cdot 100\%}{5 \cdot v_{loss}(k-i-1)} < 1\% \tag{1}$$

The idea behind this empirical rule lies on changing to asynchronous training when the training process has become more stable. In order to measure how stable the training is at a specific epoch, we utilize a 5-window running over the percentage change of the validation loss in the last epochs (value s in 1). When the mean percentage change becomes

Algorithm 2 Empirical Rule *Strategy-Switch*

```
Input: Trainable model $m$, Training and validation data $d$, Global hyperparameters hp, Number of training epochs $e$, Percentage of epochs under All-Reduce $\alpha$
Output: Trained model $m$
 1: Deploy a set w_allred of All-Reduce workers
 2: val_window = rolling_window(6)
 3: s = $+\infty$
 4: i = 0
 5: while $s > 1\%$ do
 6:     train(w_allred, m, d, hp, i)
 7:     roll(val_window, epoch_val_loss)
 8:     $s = \frac{\sum_{j=0}^{4} \|v_{loss}(i-j) - v_{loss}(i-j-1)\| \cdot 100\%}{5 \cdot v_{loss}(i-j-1)}$
 9:     i = i + 1
10: end while
11: start_epoch_ps = $i$
12: Dump m on DFS
13: Initiate a set ps of parameter servers
14: ps.load(m)
15: Deploy a set w_ps of Parameter Server workers
16: for $i = start\_epoch\_ps$ to $e$ do
17:     train(w_ps, ps, m, d, hp, i)
18: end for
19: return $m$
```

small and less than 1%, we consider this epoch a good switching point. Both the window size and the percentage threshold were found empirically. Larger window sizes were found sensitive to previous epochs with larger values of validation loss, leading the training to switch to asynchronous mode at the very last few epochs. Smaller window sizes might lead to earlier switching points, affected by smaller loss change between less successive epochs. The choice of the 1% threshold is further explained in Section IV-B.

Strategy-Switch enriched with the empirical rule (ER-SS) is described in Algorithm 2.

IV. EXPERIMENTAL EVALUATION

In this section, we present a detailed experimental evaluation on *Strategy-Switch*, compared to *All-Reduce* and *Parameter Server* training on homogeneous and heterogeneous clusters. While the experiments focus on ResNet-based architectures for CIFAR datasets, the proposed approach is not restricted to ResNets. Our primary reason for choosing these networks is the wealth of published results and well-understood convergence properties, making them particularly suitable for demonstrating the effects of switching strategies.

A. EXPERIMENTAL SETUP

The clusters and benchmarks used are the same as the ones used in the benchmarking analysis in Section I. For ease of reading, #B1 and #B2 benchmarks refer to training CIFAR-10 on ResNet-20 and CIFAR-100 on ResNet-32 respectively. #C1 and #C2 refer to the homogeneous and heterogeneous

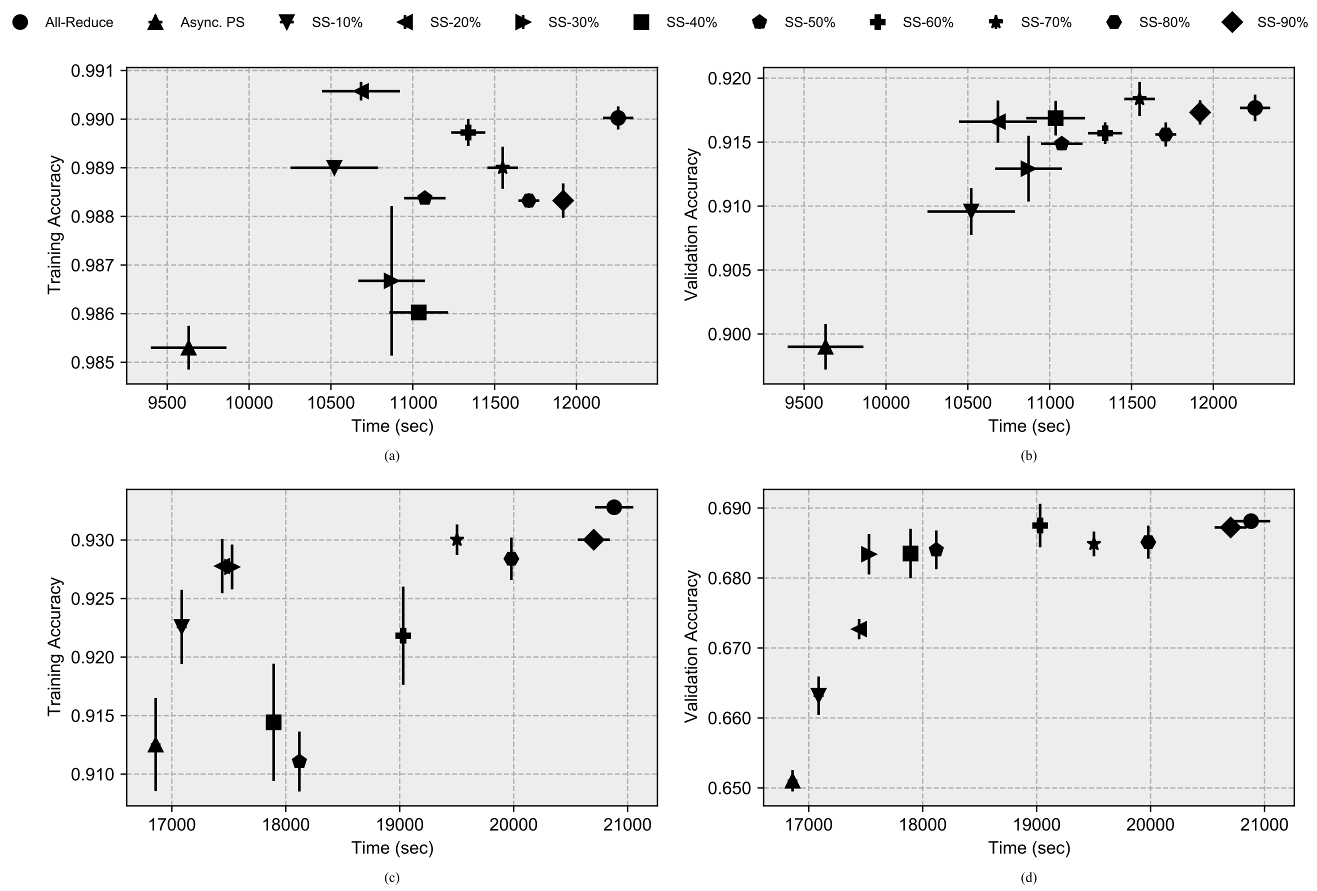

FIGURE 8. **Accuracy vs. time trade-offs in #C1 cluster for each benchmark when using *All-Reduce*, *Parameter Server* and *Strategy-Switch* with various switching points. Percentage in *Strategy-Switch* labels indicates the percentage of epochs performed under *All-Reduce* in the beginning of the training. (a) #B1 - Training Acc. (b) #B1 - Validation Acc. (c) #B2 - Training Acc. (d) #B2 - Validation Acc.**

clusters used (see Table 3 in Section II for cluster structures). As global hyperparameters, we use the ones discussed in the official ResNet paper [41] and mentioned in Table 1. We have executed every experiment 3 times and error bars outline the statistical information (i.e., average and min/max values) for every experiment. The collected metrics for each experiment run were consistently close to each other, mainly due to the fact that the selection of dataset sizes and cluster resources resulted in sufficient execution times, minimizing statistical errors that may occur when the execution times are negligible. In sections IV-B and IV-C, we discuss tradeoffs in Strategy-Switch-$\alpha\%$ and the empirical rule based on the validations loss of the *All-Reduce* training on the benchmarks in the homogeneous cluster. In sections IV-D and IV-E we evaluate, *All-Reduce*, *Parameter Server* and Empirical Rule *Strategy-Switch* on both the benchmarks and on both clusters.

B. TRADE-OFFS WHEN USING STRATEGY-SWITCH-$\alpha\%$ ON THE HOMOGENEOUS #C1 CLUSTER

Fig. 8 presents the trade-offs regarding accuracy and execution time between *All-Reduce*, *Parameter Server* and *Strategy-Switch-$\alpha\%$* strategies in the homogeneous #C1 cluster.

As observed from Fig. 8 *All-Reduce* achieves the highest train and validation accuracy for both benchmarks #B1 and #B2 in the homogeneous #C1 cluster. This can be attributed to the synchronous nature of *All-Reduce*. Therefore, the convergence of the optimization algorithm used in the training remains the same as the one of the single node. However, *All-Reduce* appears to be the slowest approach compared to the other experiments, due to the synchronization-related overheads present throughout the training process.

Unlike *All-Reduce*, *Parameter Server* achieves the lowest accuracy for both benchmarks #B1 and #B2 in the homogeneous #C1 cluster. The lack of synchronization in the *Parameter Server* harms the convergence of the optimization algorithms, due to stale gradients mentioned in Section I-B. In further detail, when training is completed under the *Parameter Server*, model parameters might have been updated in various steps with gradients computed on outdated model parameters. On the other hand, asynchronous training gives an advantage to *Parameter Server*, compared to the synchronous approach in terms of the training speed. Each worker trains the model completely independently at its own highest pace leading to the fastest possible training in the *Parameter Server* for both benchmarks #B1 and #B2.

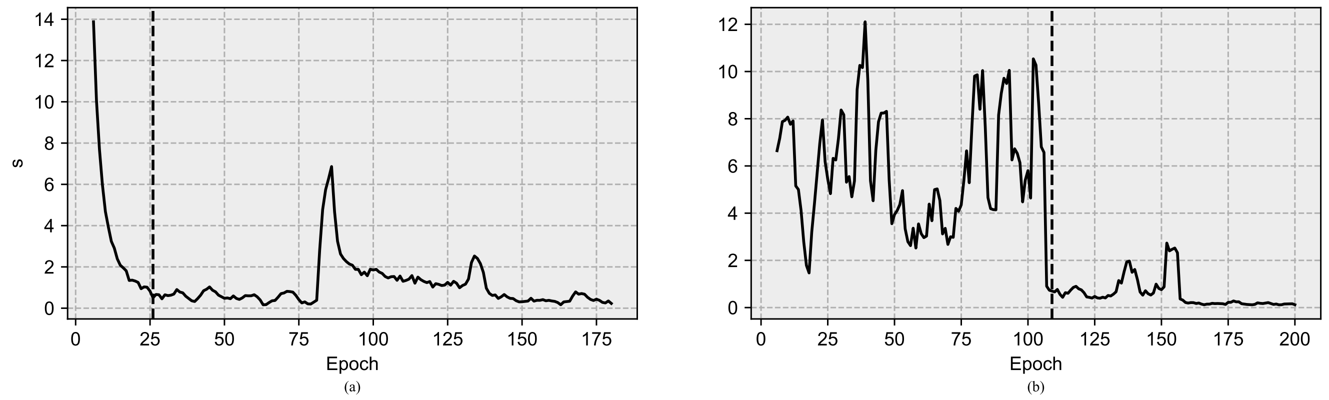

FIGURE 9. Value used in the empirical rule per training epoch on ***All-Reduce*** setups for both benchmarks. Vertical dashed line indicates the switching point according to the empirical rule. (a) #B1 - Validation Loss (b) #B2 - Validation Loss.

Strategy-Switch-$\alpha\%$ proposed in this paper, achieves a balance between *All-Reduce* and *Parameter Server* by exploiting the advantages of both strategies. In *Strategy-Switch*, the training of the model starts with the *All-Reduce* and ends with the *Parameter Server* strategy. Therefore, the first $\alpha\%$ slower epochs following the *All-Reduce* paradigm lead the model to a set of parameters closer to the optimization point, which is less prone to stale gradients of the *Parameter Server* training of the last epochs. Thus, similar levels of accuracy to *All-Reduce* can be achieved faster. α is a hyperparameter in *Strategy-Switch*, which we study by tuning α in the range $[10, 90]$ with step 10. Irrespective of the value of α, Fig. 8 indicates that, in *Strategy-Switch*, the accuracy metrics are higher than the *Parameter Server* and the training time is shorter compared to *All-Reduce*. Furthermore, the larger the value of α, the slower the training process is, due to more epochs performed under *All-Reduce*. In general, larger values of α indicate models that approach the convergence point of the *All-Reduce* training.

C. EXPLAINING THE S VALUE OF THE EMPIRICAL RULE ON THE HOMOGENEOUS #C1 CLUSTER

As explained in Section IV-B, α is a hyperparameter in *Strategy-Switch*. To identify a proper switching point, we propose the empirical rule discussed in Section III-C, leading to Empirical Rule - *Strategy-Switch*. In this section, we discuss the evolution of the s value of the empirical rule per training epoch. In Fig. 9 the s value is presented per train epoch on the *All-Reduce* training for both benchmarks. The vertical dashed line indicates the switching point according to the empirical rule.

Fig. 9a outlines the evolution of the s value for the #B1 benchmark. The validation loss drops to lower values rapidly and is stabilized early in the training process. Specifically, in epoch 26, the s value appears to satisfy the threshold of 1% in the empirical rule presented in Equation 1 of Section III-C. For ~50 epochs, the value of s is at the same levels, presenting some spikes later on some epochs. Larger s values indicate larger validation loss change. Attributed to the tuning of the learning rate (Table 1), which is decreased at epoch 81, the validation loss presents a larger variation at this point. However, the earlier steady state of the model renders epoch 26 a good switching point, since the spikes attributed to the decrease of learning rate in the s value, are smoothed after a few epochs.

Similar observations are made in Fig. 9b regarding benchmark #B2. However, the training appears to be stabilized at epoch 109, since benchmark #B2 is more complex to converge compared to benchmark #B1.

In the following sections IV-D and IV-E we evaluate Empirical Rule - *Strategy-Switch* in contrast with *All-Reduce* and *Parameter Server* for both benchmarks in the homogeneous #C1 and the heterogeneous #C2 clusters respectively.

D. STRATEGY-SWITCH *IN THE HOMOGENEOUS #C1 CLUSTER USING THE EMPIRICAL RULE.*

Fig. 10, Fig. 11 and Fig. 12 present the results of *Strategy-Switch* in the homogeneous #C1 cluster when using the empirical rule and compares them with *All-Reduce* and *Parameter Server* training. It can be observed that in both benchmarks #B1 and #B2 *Strategy-Switch* has the best trade-off between accuracy and execution time.

When reaching convergence, training, and validation accuracy are shown in Fig. 11a and Fig. 11b. Training accuracy presents almost identical values between *All-Reduce* and *Strategy-Switch*, while smaller values are observed under *Parameter Server*. The same patterns are also identified regarding validation accuracy under convergence. Fig. 10a and 10b present the execution time of all distributed training approaches for benchmarks #B1 and #B2 in the homogeneous #C1 cluster. *Parameter Server* strategy is the fastest one as expected. *Strategy-Switch* is clearly faster than *All-Reduce* strategy for both benchmarks #B1 and #B2. In further detail, for the #B1 benchmark, the converged *Strategy-Switch* model presents only a loss of 0.05% and 0.1% in the training and

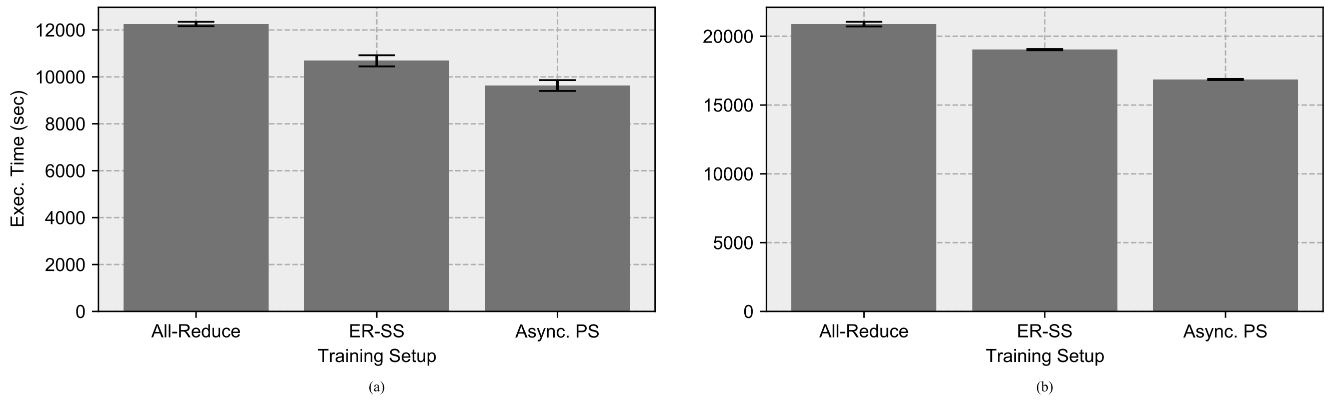

FIGURE 10. Training time under ***All-Reduce***, ***Strategy-Switch*** and ***Parameter Server*** in the homogenous #C1 cluster (a) #B1 benchmark (b) #B2 benchmark.

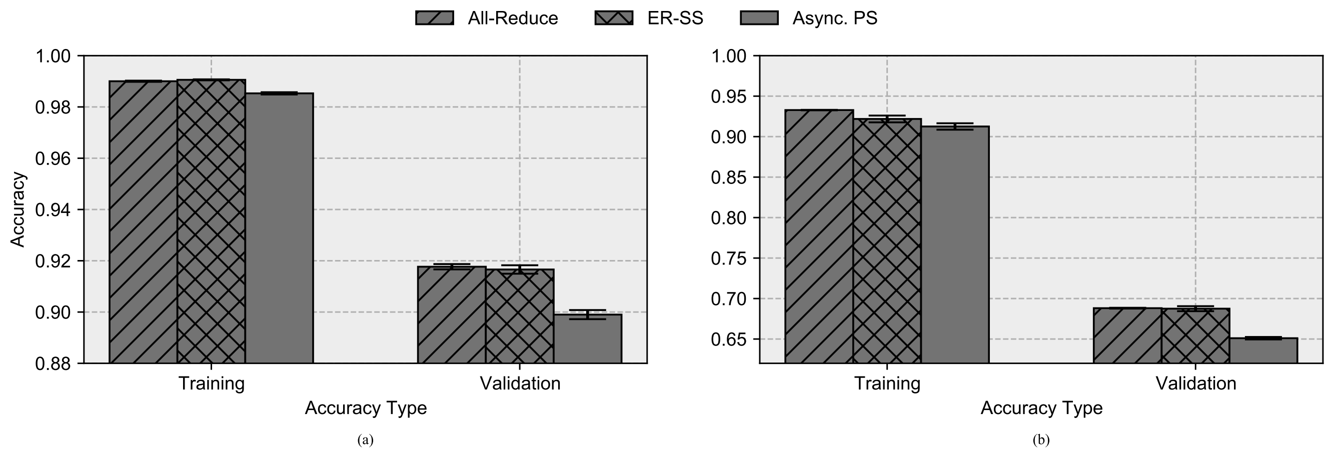

FIGURE 11. Training and validation accuracy values at convergence, when training in the homogenous #C1 cluster for the benchmarks (a) #B1 benchmark (b) #B2 benchmark.

validation accuracy respectively compared to the *All-Reduce* model, while it is trained 1.14X faster. Regarding the #B2 benchmark, with a loss of 1% and 0.06% in the training and validation accuracy, the resulting model in *Strategy-Switch* is training with 1.1X speedup.

Fig. 12c and 12d outline the training and validation accuracy at each epoch during training the benchmarks #B1 and #B2. It can be observed that when training under *Strategy-Switch*, training loss follows the same pattern as the case of training under *All-Reduce*, while in *Parameter Server* the loss presents a different evolution. The training setups have the same behavior regarding validation accuracy, as further outlined in Fig. 12. The evolution of the training and validation loss across epochs for benchmarks #B1 and #B2 are illustrated in Fig. 12a and 12b. It is important to note that validation loss under *Strategy-Switch* evolves into smaller values across epochs compared to *All-Reduce* and *Parameter Server*. Regarding training loss, under *Strategy-Switch* we achieve similar trends to *Parameter Server* training, which evolves better compared to *All-Reduce*.

E. STRATEGY-SWITCH *IN THE HETEROGENEOUS #C2 CLUSTER USING THE EMPIRICAL RULE*

Fig. 13, Fig. 14 and Fig. 15 present the results of *Strategy-Switch* when using the empirical rule in the heterogeneous #C2 cluster in comparison with results from models trained under *All-Reduce* and *Parameter Server*. As in Section IV-D, *Strategy-Switch* presents the best trade-offs between the two baseline distributed approaches.

Fig. 15 indicates similar trends in the evolution of loss and accuracy metrics to the ones derived from training in the homogeneous cluster. Specifically, *Strategy-Switch* models follow similar trends to the ones of the *All-Reduce* models regarding accuracy. In terms of training and validation loss, *Strategy-Switch* finally reaches lower levels compared to the other training setups.

The greatest difference between the results of the heterogeneous #C2 cluster and the homogeneous #C1 cluster is related to the execution time. Let us discuss in further detail the trade-offs between accuracy and training time in the heterogeneous #C2 cluster. For the benchmark #B1, Fig. 13a

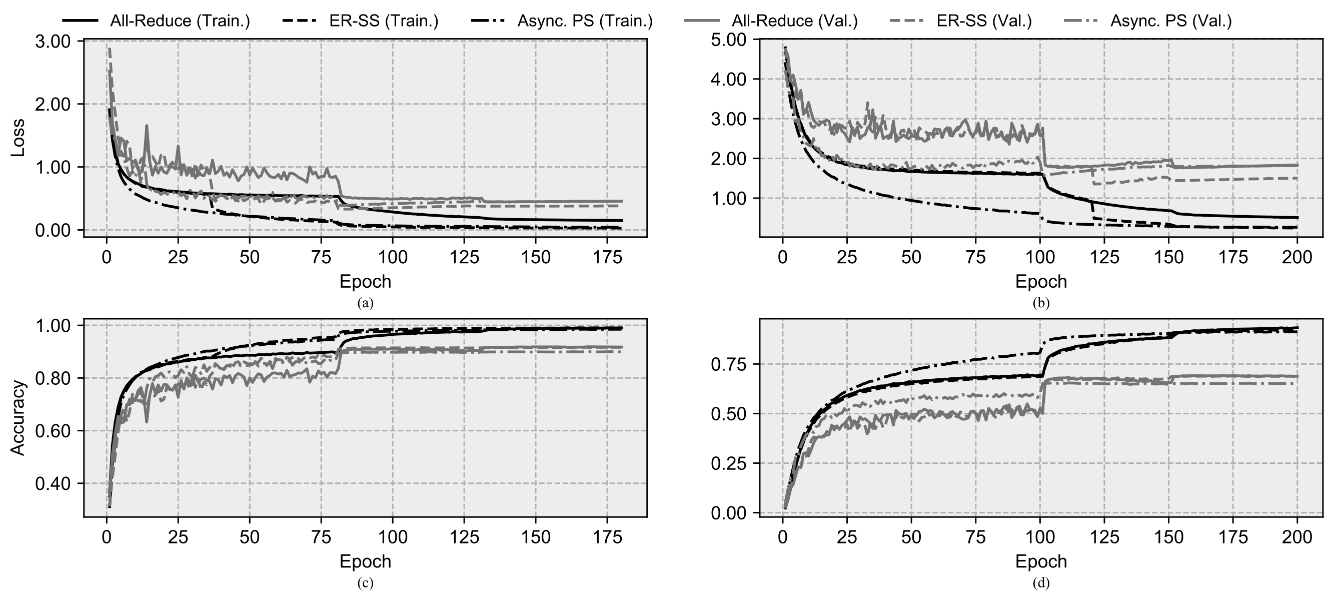

FIGURE 12. Loss and accuracy, when training in the homogenous #C1 cluster for the benchmarks. Black lines indicate training metrics and gray lines indicate validation metrics. (a) Loss - #B1 benchmark (b) Loss - #B2 benchmark (c) Accuracy - #B1 benchmark (d) Accuracy - #B2 benchmark.

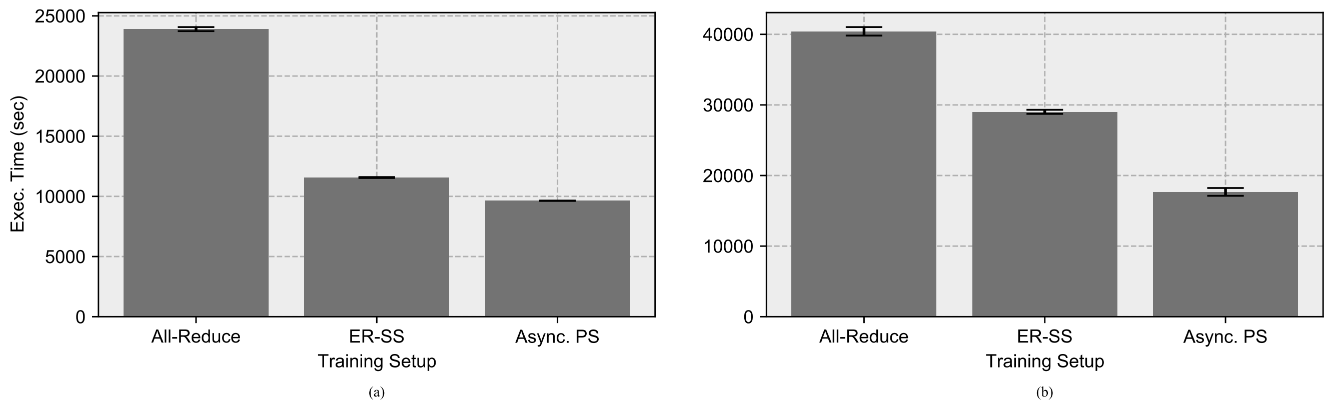

FIGURE 13. Training time under *All-Reduce*, *Strategy-Switch* and *Parameter Server* in the heterogeneous #C2 cluster (a) #B1 benchmark (b) #B2 benchmark.

and Fig. 14a present the execution time and resulting accuracy values when training models under the three distributed approaches. *Strategy-Switch* completes the training 2.07*X* faster compared to *All-Reduce*. *Strategy-Switch* also appears to present the greatest value in both training and validation accuracy (∼0.05% greater than *All-Reduce*). In *Strategy-Switch* the model parameters are initialized by *All-Reduce* for the rest of the training to be performed under the *Parameter Server*. Due to the heterogeneity of the cluster and the lack of synchronization in the *Parameter Server*, the faster workers will dominate the training until they finish, leading to fewer stale parameters in the slow machines. On the contrary, the slow machines will continue the train at their own pace when the faster machines have finished, without the stale gradients effect. This observation can explain the slight increase in the accuracy values in *Strategy-Switch*. Similar trends are observed in the #B2 benchmark. In this case, execution time and accuracy values are provided in Fig. 13b and 14b respectively. The model derived from *Strategy-Switch* is created 1.4*X* faster than the *All-Reduce* one, with a validation accuracy slightly enhanced by 0.19%. In both benchmarks *Parameter Server* is the faster (2.48*X* for #B1 and 2.28 for #B2 speedup compared to *All-Reduce*), but lacks in models quality (validation accuracy harmed by 0.9% for #B1 and 2.88% for #B2 compared to *All-Reduce*).

V. RELATED WORK

As discussed in Section III, *Strategy-Switch* is inspired from *Sync-Switch* [37], by changing the BSP *Parameter Server* to *All-Reduce* training, in the first epochs of the hybrid training. However, we do not compare our approach to *Sync-Switch*. The reason behind this choice is that the two

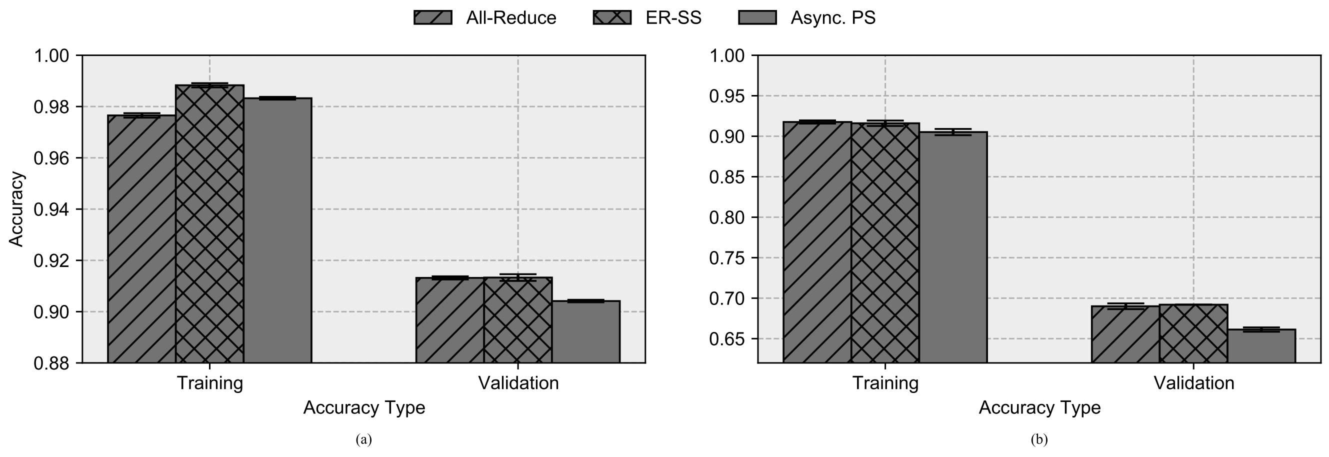

FIGURE 14. Training and validation accuracy values at convergence, when training in the heterogeneous #C2 cluster for the benchmarks (a) #B1 benchmark (b) #B2 benchmark.

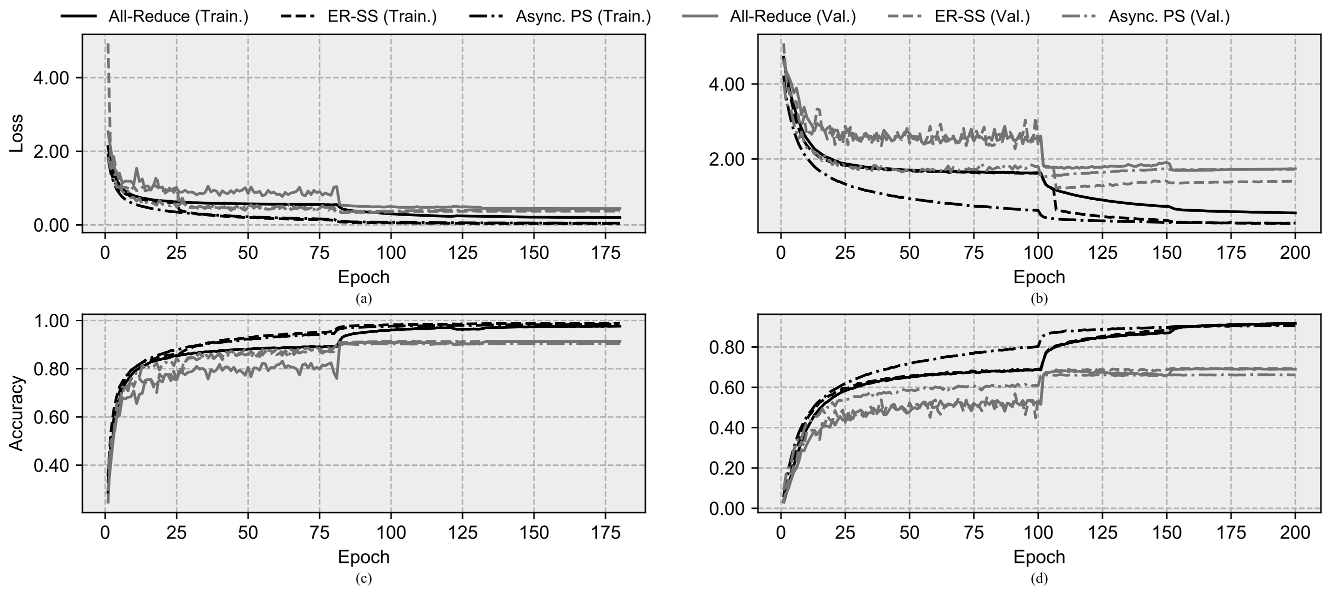

FIGURE 15. Loss and accuracy, when training in the heterogeneous #C2 cluster for the benchmarks. Black lines indicate training metrics and gray lines indicate validation metrics. (a) Loss - #B1 benchmark (b) Loss - #B2 benchmark (c) Accuracy - #B1 benchmark (d) Accuracy - #B2 benchmark.

approaches identify the switching point in different setups. in *Sync-Switch*, the authors propose an offline switching point policy based on a binary search on the converged accuracy of various runs. In *Strategy-Switch*, we propose an empirical rule applied to the synchronous *All-Reduce* online. Therefore we could not compare to their offline approach that needs multiple runs. In *Sync-Switch* they also propose an online protocol for transient stragglers that might appear and disappear throughout the training. Since we benchmark on clearly homogeneous and heterogeneous setups without transient stragglers, their is no direct comparison between our methods. In homogeneous setups, their online protocol could result in ASP training, while in our heterogeneous environment with constant straggler VMs *Sync-Switch* would result in BSP *Parameter Server*.

A comprehensive survey on distributed machine learning is presented in [46] (Figure 3 discusses about different ML topologies). An updated survey on decentralized federated learning is presented in [47]. Although we do not place *Strategy-Switch* in the category of federated learning, we borrow methodologies from it during model exchanges, etc. In [48] the authors present Epidemic Learning, a distributed approach where participating nodes randomly exchange model updates. The authors present theoretical guarantees and experimental evaluation that indicate they can converge quicker and with a slightly better accuracy than baselines. In PANM [49] the authors also exploit decentralization in a federated setting to achieve efficient training by detecting node clusters, showcasing that a distributed approach where weights are exchanged between peers can be beneficial. The authors of [50] evaluate numerous autonomous connection methodologies in peer to peer settings and showcase that the the connection mechanism plays a crucial role in the model accuracy and convergence time.

A. CONSISTENCY CONTROL VS. TRAINING TIME

Recently, there have been multiple works trying to enhance both the *All-Reduce* and the *Parameter Server* training. For example, Prague [51] was proposed as an asynchronous *All-Reduce* alternative exploiting primitives of AD-PSGD [52] for faster model training in heterogeneous environments. In *Parameter Server*, there have been proposed multiple research papers discussing trade-offs between consistency and performance, with SSP [42] as the most prominent, and DSSP [53] as an evolution with dynamic approved staleness values. In fedPAGE [54] the authors adaptively prune workers according to their capabilities (i.e., networking bandwidth, CPU processing power, etc.) until all workers achieve a balanced "speed". FEDL [55] balances local computation rounds (i.e., local weight updates) with global communication rounds (i.e., model exchanges) to optimize convergence rate and accuracy. PFL [56] also adapts local and global computation timing and exchange between peers to optimize the entire model convergence efficiency, by employing a set of algorithmic approaches. PruneFL [57] adaptively prunes the model size according to the capabilities of each worker while. Such works sacrifice less model quality in respect with execution time, but *Strategy-Switch* appears to present similar metrics to the synchronous approach.

B. OTHER OPTIMIZATIONS

For heterogeneous environments, there have been proposed various hyperparameter adjustments, such as learning rate [53] and mini-batch size adjustments [58] per worker. Regarding network utilization, there have been also multiple optimizations proposed including gradient sparsification [32], [33], [59], [60] and quantization [61], [62], [63], [64]. For parameter server training, the network could also be optimized via parameter management in servers [65], [66]. Such optimizations are orthogonal to *Strategy-Switch* and could be applied for further optimizations.

VI. CONCLUSION AND FUTURE WORK

In this work, we propose *Strategy-Switch* as a hybrid distributed training approach exploiting *All-Reduce* and asynchronous *Parameter Server* for more efficient neural network training. We also enrich *Strategy-Switch* with an empirical rule, that can decide the switching point from *All-Reduce* to asynchronous *Parameter Server* for best model metrics similar to the *All-Reduce*'s ones. Our experimental evaluation indicated the same quality models as the ones derived from synchronous approaches with up to 1.14X speedup. On heterogeneous environments, *Strategy-Switch* managed to create models with slightly enhanced accuracy metrics on training and validation sets by up to 2.07X faster.

Although we have focused on CIFAR-10/100 benchmarks and ResNet-based architectures, the proposed scheme can be extended to other tasks and architectures that exhibit typical deep-learning convergence patterns. For more comprehensive coverage, future work includes an exploration of alternative model families (e.g., Vision Transformers, NLP-focused architectures), scenarios where transient stragglers appear sporadically, and additional large-scale experiments involving random capacity distributions. We also aim to refine and mathematically analyze the empirical rule's hyperparameters (rolling window size and threshold), investigating whether an adaptive strategy or theoretically guided threshold could further optimize switching performance. Finally, we plan to incorporate GPU-based experiments and advanced synchronization schemes, highlighting how *Strategy-Switch* could extend into settings with more powerful hardware accelerators.

ACKNOWLEDGMENT

The work was done while Iasonas Chalas was with the School of Electrical and Computer Engineering, National Technical University of Athens, Greece.

REFERENCES

[1] N. Provatas, "Exploiting data distribution in distributed learning of deep classification models under the parameter server architecture," in *Proc. VLDB PhD Workshop (VLDB-PhD)*, Copenhagen, Denmark, Jan. 2021, pp. 1–4.

[2] R. Socher, B. Huval, B. Bath, C. D. Manning, and A. Ng, "Convolutional-recursive deep learning for 3D object classification," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, pp. 1–9.

[3] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, "Residual attention network for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6450–6458.

[4] X. Yang, Y. Ye, X. Li, R. Y. K. Lau, X. Zhang, and X. Huang, "Hyperspectral image classification with deep learning models," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 9, pp. 5408–5423, Sep. 2018.

[5] R. Almajalid, J. Shan, Y. Du, and M. Zhang, "Development of a deep-learning-based method for breast ultrasound image segmentation," in *Proc. 17th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2018, pp. 1103–1108.

[6] S. Ghosh, N. Das, I. Das, and U. Maulik, "Understanding deep learning techniques for image segmentation," *ACM Comput. Surveys*, vol. 52, no. 4, pp. 1–35, Jul. 2020.

[7] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 7, pp. 3523–3542, Jul. 2022.

[8] J. Huang and B. Kingsbury, "Audio-visual deep learning for noise robust speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 7596–7599.

[9] Z. Zhang, J. Geiger, J. Pohjalainen, A. E.-D. Mousa, W. Jin, and B. Schuller, "Deep learning for environmentally robust speech recognition: An overview of recent developments," *ACM Trans. Intell. Syst. Technol.*, vol. 9, no. 5, pp. 1–28, Sep. 2018.

[10] S. Petridis, Z. Li, and M. Pantic, "End-to-end visual speech recognition with LSTMS," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2017, pp. 2592–2596.

[11] K. Kowsari, D. E. Brown, M. Heidarysafa, K. Jafari Meimandi, M. S. Gerber, and L. E. Barnes, "HDLTex: Hierarchical deep learning for text classification," in *Proc. 16th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2017, pp. 364–371.

[12] J. Liu, W.-C. Chang, Y. Wu, and Y. Yang, "Deep learning for extreme multi-label text classification," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Aug. 2017, pp. 115–124.

[13] C. N. Kamath, S. S. Bukhari, and A. Dengel, "Comparative study between traditional machine learning and deep learning approaches for text classification," in *Proc. ACM Symp. Document Eng.*, Aug. 2018, pp. 1–11.

[14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. neural Inf. Process. Syst.*, vol. 25, 2012, pp. 1–9.

[16] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[17] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, Jan. 2015, pp. 448–456.

[18] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.

[19] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. Van Der Maaten, "Exploring the limits of weakly supervised pretraining," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Jan. 2018, pp. 185–201.

[20] J. Yu, Z. Wang, V. Vasudevan, L. Yeung, M. Seyedhosseini, and Y. Wu, "CoCa: Contrastive captioners are image-text foundation models," 2022, *arXiv:2205.01917*.

[21] X.-W. Chen and X. Lin, "Big data deep learning: Challenges and perspectives," *IEEE Access*, vol. 2, pp. 514–525, 2014.

[22] E. Kassela, N. Provatas, I. Konstantinou, A. Floratou, and N. Koziris, "General-purpose vs. specialized data analytics systems: A game of ML & SQL thrones," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2019, pp. 317–326.

[23] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst. 32*, Jan. 2019, pp. 8024–8035.

[24] E. P. Xing, Q. Ho, W. Dai, J. K. Kim, J. Wei, S. Lee, X. Zheng, P. Xie, A. Kumar, and Y. Yu, "Petuum: A new platform for distributed machine learning on big data," *IEEE Trans. Big Data*, vol. 1, no. 2, pp. 49–67, Jun. 2015.

[25] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, "MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems," 2015, *arXiv:1512.01274*.

[26] M. Abadi et al., "{TensorFlow}: A system for {Large–Scale} machine learning," in *Proc. 12th USENIX Symp. operating Syst. design Implement. (OSDI)*, 2016, pp. 265–283.

[27] M. Cho, U. Finkler, and D. Kung, "Blueconnect: Novel hierarchical all-reduce on multi-tired network for deep learning," in *Proc. 2nd SysML Conf.*, 2019, pp. 1–8.

[28] X. Miao, X. Nie, Y. Shao, Z. Yang, J. Jiang, L. Ma, and B. Cui, "Heterogeneity-aware distributed machine learning training via partial reduce," in *Proc. Int. Conf. Manage. Data*, Jun. 2021, pp. 2262–2270.

[29] P. Patarasuk and X. Yuan, "Bandwidth optimal all-reduce algorithms for clusters of workstations," *J. Parallel Distrib. Comput.*, vol. 69, no. 2, pp. 117–124, Feb. 2009.

[30] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, Q. Le, and A. Ng, "Large scale distributed deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25. Newry, NIR, U.K.: Curran Associates, 2012, pp. 1–9.

[31] A. Smola and S. Narayanamurthy, "An architecture for parallel topic models," *Proc. VLDB Endowment*, vol. 3, nos. 1–2, pp. 703–710, Sep. 2010.

[32] M. Li, L. Zhou, Z. Yang, A. Li, F. Xia, D. G. Andersen, and A. Smola, "Parameter server for distributed machine learning," in *Proc. Big Learn. NIPS workshop*, vol. 6, 2013, p. 2.

[33] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server," in *Proc. 11th USENIX Symp. Operating Syst. Design Implement. (OSDI)*, 2014, pp. 583–598.

[34] G. Wang, S. Venkataraman, A. Phanishayee, J. Thelin, N. R. Devanur, and I. Stoica, "Blink: Fast and generic collectives for distributed ML," in *Proc. Mach. Learn. Syst.*, Jan. 2019, pp. 172–186.

[35] Y. Lu, H. Gu, X. Yu, and K. Chakrabarty, "Lotus: A new topology for large-scale distributed machine learning," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 17, no. 1, pp. 1–21, Jan. 2021.

[36] N. Provatas, I. Konstantinou, and N. Koziris, "Is systematic data sharding able to stabilize asynchronous parameter server training?" in *Proc. 2021 IEEE Int. Conf. Big Data (Big Data)*, 2021, pp. 1092–1101.

[37] S. Li, O. Mangoubi, L. Xu, and T. Guo, "Sync-switch: Hybrid parameter synchronization for distributed deep learning," in *Proc. IEEE 41st Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2021, pp. 528–538.

[38] R. Thakur, R. Rabenseifner, and W. Gropp, "Optimization of collective communication operations in MPICH," *Int. J. High Perform. Comput. Appl.*, vol. 19, no. 1, pp. 49–66, Feb. 2005.

[39] S. Gupta, W. Zhang, and F. Wang, "Model accuracy and runtime tradeoff in distributed deep learning: A systematic study," in *Proc. IEEE 16th Int. Conf. Data Mining (ICDM)*, Dec. 2016, pp. 171–180.

[40] A. Krizhevsky, "Learning multiple layers of features from tiny images," Master's thesis, Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, 2009.

[41] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[42] Q. Ho, J. Cipar, H. Cui, J. K. Kim, S. Lee, P. B. Gibbons, G. A. Gibson, G. R. Ganger, and E. P. Xing, "More effective distributed ML via a stale synchronous parallel parameter server," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2013, pp. 1223–1231.

[43] Y. Li, M. Yu, S. Li, S. Avestimehr, N. S. Kim, and A. G. Schwing, "Pipe-SGD: A decentralized pipelined SGD framework for distributed deep net training," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, Dec. 2018, pp. 8056–8067.

[44] F. N. Iandola, M. W. Moskewicz, K. Ashraf, and K. Keutzer, "FireCaffe: Near-linear acceleration of deep neural network training on compute clusters," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2592–2600.

[45] S. Shi, Z. Tang, X. Chu, C. Liu, W. Wang, and B. Li, "A quantitative survey of communication optimizations in distributed deep learning," *IEEE Netw.*, vol. 35, no. 3, pp. 230–237, May 2021.

[46] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, "A survey on distributed machine learning," *ACM Comput. Surv.*, vol. 53, no. 2, pp. 1–33, Mar. 2020.

[47] L. Yuan, Z. Wang, L. Sun, P. S. Yu, and C. G. Brinton, "Decentralized federated learning: A survey and perspective," *IEEE Internet Things J.*, vol. 11, no. 21, pp. 34617–34638, Nov. 2024.

[48] M. D. Vos, S. Farhadkhani, R. Guerraoui, A.-M. Kermarrec, R. Pires, and R. Sharma, "Epidemic learning: Boosting decentralized learning with randomized communication," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 36, Jan. 2024, pp. 1–33.

[49] Z. Li, J. Lu, S. Luo, D. Zhu, Y. Shao, Y. Li, Z. Zhang, Y. Wang, and C. Wu, "Towards effective clustered federated learning: A peer-to-peer framework with adaptive neighbor matching," *IEEE Trans. Big Data*, vol. 10, no. 6, pp. 812–826, Dec. 2022.

[50] R. Šajina, N. Tanković, and I. Ipšić, "An overview of autonomous connection establishment methods in peer-to-peer deep learning," *IEEE Access*, vol. 12, pp. 111752–111768, 2024.

[51] Q. Luo, J. He, Y. Zhuo, and X. Qian, "Prague: High-performance heterogeneity-aware asynchronous decentralized training," in *Proc. 25th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Mar. 2020, pp. 401–416.

[52] X. Lian, W. Zhang, C. Zhang, and J. Liu, "Asynchronous decentralized parallel stochastic gradient descent," in *Proc. Int. Conf. Mach. Learn.*, Jan. 2017, pp. 3043–3052.

[53] X. Zhao, A. An, J. Liu, and B. X. Chen, "Dynamic stale synchronous parallel distributed training for deep learning," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2019, pp. 1507–1517.

[54] G. Zhou, Q. Li, Y. Liu, Y. Zhao, Q. Tan, S. Yao, and K. Xu, "FedPAGE: Pruning adaptively toward global efficiency of heterogeneous federated learning," *IEEE/ACM Trans. Netw.*, vol. 32, no. 3, pp. 1873–1887, Jun. 2024.

[55] C. T. Dinh, N. H. Tran, M. N. H. Nguyen, C. S. Hong, W. Bao, A. Y. Zomaya, and V. Gramoli, "Federated learning over wireless networks: Convergence analysis and resource allocation," *IEEE/ACM Trans. Netw.*, vol. 29, no. 1, pp. 398–409, Feb. 2021.

[56] J. Lu, H. Liu, R. Jia, J. Wang, L. Sun, and S. Wan, "Towards personalized federated learning via group collaboration in IIoT," *IEEE Trans. Ind. Informat.*, vol. 19, no. 8, pp. 8923–8932, Nov. 2023.

[57] Y. Jiang, S. Wang, V. Valls, B. J. Ko, W.-H. Lee, K. K. Leung, and L. Tassiulas, "Model pruning enables efficient federated learning on edge devices," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 12, pp. 10374–10386, Apr. 2023.

[58] C. Chen, Q. Weng, W. Wang, B. Li, and B. Li, "Fast distributed deep learning via worker-adaptive batch sizing," in *Proc. ACM Symp. Cloud Comput.*, Oct. 2018, p. 521.

[59] J. Zhou, X. Li, P. Zhao, C. Chen, L. Li, X. Yang, Q. Cui, J. Yu, X. Chen, Y. Ding, and Y. A. Qi, "KunPeng: Parameter server based distributed learning systems and its applications in Alibaba and ant financial," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2017, pp. 1693–1702.

[60] S. H. Hashemi, S. A. Jyothi, and R. H. Campbell, "TicTac: Accelerating distributed deep learning with communication scheduling," in *Proc. Mach. Learn. Syst.*, Jan. 2018, pp. 418–430.

[61] J. Wei, W. Dai, A. Qiao, Q. Ho, H. Cui, G. R. Ganger, P. B. Gibbons, G. A. Gibson, and E. P. Xing, "Managed communication and consistency for fast data-parallel iterative analytics," in *Proc. 6th ACM Symp. Cloud Comput.*, Aug. 2015, pp. 381–394.

[62] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li, "TernGrad: Ternary gradients to reduce communication in distributed deep learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, Dec. 2017, pp. 1508–1518.

[63] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, Jan. 2016, pp. 1–12.

[64] F. Alimisis, P. M. Davies, and D. Alistarh, "Communication-efficient distributed optimization with quantized preconditioners," in *Proc. Int. Conf. Mach. Learn.*, Jan. 2021, pp. 196–206.

[65] A. Renz-Wieland, T. Drobisch, Z. Kaoudi, R. Gemulla, and V. Markl, "Just move it!: Dynamic parameter allocation in action," *Proc. VLDB Endowment*, vol. 14, no. 12, pp. 2707–2710, Jul. 2021.

[66] A. Renz-Wieland, R. Gemulla, Z. Kaoudi, and V. Markl, "NuPS: A parameter server for machine learning with non-uniform parameter access," in *Proc. Int. Conf. Manage. Data*, Jun. 2022, pp. 481–495.

NIKODIMOS PROVATAS received the Master of Engineering degree in electrical and computer engineering and the Master of Science degree in data science and machine learning from the National Technical University of Athens (NTUA), in 2016 and 2020, respectively, where he is currently pursuing the Ph.D. degree with the Computing Systems Laboratory. He was a Teaching Assistant in various big-data related courses offered by the School of Electrical and Computer Engineering, NTUA. Currently, he is a Senior-Level Data Engineer. His research interests include machine learning and big data topics.

IASONAS CHALAS received the Diploma degree (Hons.) in electrical and computer engineering from the National Technical University of Athens. He is currently pursuing the M.Sc. degree in data science with ETH Zurich.

IOANNIS KONSTANTINOU received the Diploma degree in electrical and computer engineering, the M.Sc. degree in techno-economic systems, and the Ph.D. degree from the National Technical University of Athens (NTUA), in 2004, 2007, and 2011, respectively. He is currently an Assistant Professor with the Informatics and Telecommunications Department, University of Thessaly, where he regularly teaches operating systems and programming courses. He is also a Senior Researcher with the Computing Systems Laboratory, National Technical University of Athens, where he also teaches advanced topics in databases. He currently serves as a member for the Board of Directors of KTP SA. His research interests include large scale distributed data management systems (cloud computing and big-data systems). He was a recipient of one Best Paper Award (IEEE CCGRID 2013) and one Best Paper Award Nomination (IEEE CCGRID 2015) for his work on large scale distributed systems.

NECTARIOS KOZIRIS (Member, IEEE) is currently a Professor in computer science and the Dean of the School of Electrical and Computer Engineering, National Technical University of Athens. His research interests include parallel and distributed systems, interaction between compilers, OS and architectures, datacenter hyperconvergence, scalable data management, and large scale storage systems. Since 1998, he has been involved in the organization of many international scientific conferences, including IPDPS, ICPP, SC, and SPAA. He has given many invited talks in conferences and universities. He was a recipient of two best paper awards for his research in parallel and distributed computing (IEEE/ACM IPDPS 2001 and CCGRID 2013) and had received honorary recognition from Intel, in 2015, for his research and insightful contributions in transactional memory (TSX synchronization extensions). He has participated as a Partner or Consortium Coordinator in several EU projects involving large-scale systems (ACTiCLOUD, EuroEXA, SELIS, CELAR, ASAP, EGI, PRACE, GREDIA, GRID4ALL, and ARCOMEM). He is a member of the IEEE Computer Society, a Senior Member of the ACM, and the elected Chair of the IEEE Greece Section and started the IEEE Computer Society Greece. To promote the open source software in Greece, he co-founded the Greek Free/Open Source Software Society (GFOSS-www.ellak.gr), in 2008, with members 29 Greek Universities and Research Centers, where he is also serving as the Vice-Chair for the Board of Directors.