ASAP.SGD: Instance-based Adaptiveness to Staleness in Asynchronous SGD

Karl Bäckström [1] Marina Papatriantafilou [1] Philippas Tsigas [1]

Abstract

Concurrent algorithmic implementations of Stochastic Gradient Descent (SGD) give rise to critical questions for compute-intensive Machine Learning (ML). Asynchrony implies speedup in some contexts, and challenges in others, as stale updates may lead to slower, or non-converging executions. While previous works showed asynchrony-adaptiveness can improve stability and speedup by reducing the step size for stale updates according to static rules, there is no one-size-fits-all adaptation rule, since the optimal strategy depends on several factors. We introduce (i) ASAP.SGD, an analytical framework capturing necessary and desired properties of staleness-adaptive step size functions and (ii) TAIL-τ, a method for utilizing key properties of the *execution instance*, generating a tailored strategy that not only dampens the impact of stale updates, but also leverages fresh ones. We recover convergence bounds for adaptiveness functions satisfying the ASAP.SGD conditions, for general, convex and non-convex problems, and establish novel bounds for ones satisfying the Polyak-Lojasiewicz property. We evaluate TAIL-τ with representative *AsyncSGD* concurrent algorithms, for Deep Learning problems, showing TAIL-τ is a vital complement to *AsyncSGD*, with (i) persistent speedup in wall-clock convergence time in the parallelism spectrum, (ii) considerably lower risk of non-convergence, as well as (iii) precision levels for which original SGD implementations fail.

1. Introduction

The ascending interest in concurrent SGD is due to the explosion of data volumes, requiring scalable systems to process them in ML and Artificial Neural Network (ANN) applications. However, parallelization of the inherently sequential SGD process is non-trivial since each iteration requires the computation of the previous one. Besides understanding the dynamics of such executions, achieving resource-efficiency is a known significant target, since it can imply significant improvements in energy efficiency.

Traditional synchronous SGD (*SyncSGD*) conforms to the sequential SGD semantics by employing iteration-level parallelism, and lock-step-style synchronization. In practice, *SyncSGD* accelerates updates by data-parallel concurrent gradient computation, e.g. by GPU-acceleration, or aggregating gradient contributions in distributed settings. *SyncSGD* improves computational efficiency up to a point, but suffers direct limitations, as each iteration is at least as slow as the slowest contributing worker. However, from an optimization standpoint *SyncSGD* is analogous to sequential SGD, with well-understood convergence properties.

In contrast, asynchronous concurrent SGD (*AsyncSGD*) introduces a higher-level parallelism by relaxing the sequential SGD semantics, allowing asynchronous reads/updates on the shared ML model θ. Consequently, *AsyncSGD* provides computational benefits, however at the price of *asynchrony-induced noise* due to the *staleness* τ that arises when updates are not applied to the same states based on which they were computed, but instead on ones that have been updated τ times in-between. It was within *convex optimization*, targeting primarily regression problems (Zinkevich et al., 2009; 2010), where it was shown that the *asynchrony-induced noise* had small impact on the quality of the updates, and that the computational benefits of reduced synchronization provided speedup for certain problems. A relevant example is HOGWILD! which, with only component-wise atomic access to θ (i.e. relaxed consistency by not ensuring atomicity for the complete vector) showed almost-linear speedup for strongly convex and sparse problems (Recht et al., 2011; Alistarh et al., 2018; Nguyen et al., 2018). Asymptotic bounds for *AsyncSGD* were similarly established under strong convexity and smoothness assumptions, and for non-convex problems, such as matrix completion (De Sa et al., 2015).

However, these analytical confinements make the concluding outcomes non-applicable for a wider class of applications, including Deep Learning (DL), characterized by inherent non-convexity. This is confirmed in recent works (Wei et al., 2019; Bäckström et al., 2019; Lopez et al., 2020; Bäck-

[1]Department of Computer Science and Engineering, Chalmers University of Technology, Gothenburg, Sweden. Correspondence to: Karl Bäckström <bakarl@chalmers.se>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

ström et al., 2021), showing challenges in achieving stable and high-quality convergence with *AsyncSGD* for ANN training, due to staleness. The importance of progress and consistency guarantees of *AsyncSGD* is emphasized in (Bäckström et al., 2021), where the introduced lock-free and consistent *Leashed-SGD* shows major improvements in convergence stability (reduced risk of non-convergence) compared to lock-based *AsyncSGD* and HOGWILD!.

Recent works show that *asynchrony-awareness* can reduce negative effects of staleness, by dampening the step size of stale updates (Sra et al., 2016; Zhang et al., 2016; Bäckström et al., 2019; Ren et al., 2020; Damaskinos et al., 2020). In particular, staleness-adaptiveness has been proven to reduce the statistical penalty of asynchrony in *AsyncSGD*, thereby improving its stability and convergence rate.

Challenges. In summary, stable convergence of *AsyncSGD* is critically sensitive to (i) parallelism degree, (ii) asynchrony-awareness, and (iii) progress and consistency guarantees of the algorithmic implementation, and the mechanisms to ensure them, e.g. locking. These factors have been studied mostly in isolation, and there is an imminent need to evaluate them in conjunction, to understand how *AsyncSGD* can be utilized effectively in practice.

Moreover, existing staleness-adaptive methods either (i) statically scale the overall system step size at initialization or (ii) use a pre-defined heuristic or staleness model to regulate the step size based on the observed staleness. An inherent pitfall of adapting the step size during execution is that the *overall* magnitude might be altered, which by itself will impact the efficiency. E.g., previous works employ adaptiveness strategies that almost exclusively *diminish* the overall step size (Sra et al., 2016; Zhang et al., 2016; Ren et al., 2020; Damaskinos et al., 2020). This is problematic for several reasons, e.g., (i) it may be fatal for applications sensitive to the choice of the step size (read: Deep Learning), leading to non-converging executions, and (ii) it introduces ambiguity regarding the source of potential performance improvements, reducing comparability between methods. Besides, these approaches take no consideration of the effect of underlying system parameters, such as hardware, scheduling, synchronization and consistency properties. These aspects, just like the number of workers, and other hyperparameters, significantly influence the convergence rate in general (Ma et al., 2019), and the *staleness distribution* in particular (Bäckström et al., 2021) and can even result in *multi-modal* ones as we show here. Hence, there are inherent challenges in designing adaptation schemes capable of incorporating the effects of all of these critical aspects in conjunction.

Contributions. We introduce the *instance-based asynchrony-awareness* paradigm, with the execution-dynamic TAIL-τ staleness-adaptive step size function. We also establish a framework for *<u>a</u>daptiveness to <u>s</u>taleness in <u>a</u>synchronous <u>p</u>arallel <u>SGD</u>* (ASAP.SGD), captuting key properties of such functions in general. In detail:

- ASAP.SGD captures general staleness-adaptive step size function properties, (i) necessary for maintaining overall step size magnitudes and ensuring method comparability, and (ii) desired for prioritizing gradient freshness.
- Within ASAP.SGD, we introduce TAIL-τ, a dynamic staleness-adaptive step size function, that (i) utilizes the observed *staleness distribution* as means to implicitly take underlying system parameters into consideration, and (ii) generates an execution-specific adaptation strategy, in the spirit of instance-based optimization (Kraska, 2021)
- We recover asymptotic convergence bounds for TAIL-τ in particular, and general ones within the ASAP.SGD framework, for convex and non-convex applications. We establish novel bounds for loss functions satisfying the Polyak-Lojasiewicz (PL) condition, which characterizes the shape of non-convexity, and is satisfied by several relevant ML loss functions, including least squares, logistic regression, support vector machines and certain deep ANNs.
- We implement TAIL-τ, extending existing concurrent *AsyncSGD* implementations (Bäckström, 2021), to promote further exploration of general staleness-adaptiveness within ASAP.SGD. The results show that TAIL-τ is a vital complement for *fast* and *stable* convergence for any *AsyncSGD* implementation, across the parallelism spectrum, due to its dynamic instance-based generation of tailored step size strategies. In particular, the evaluation, capturing several representative system features associated with synchronization, parallelism, execution-ordering properties, shows that for image classification training with LeNet and MLP, on MNIST, Fashion-MNIST, and CIFAR-10, TAIL-τ achieves significantly faster convergence persistently (e.g. 60% speedup, on average, for LeNet training on MNIST), for three fundamentally different *AsyncSGD* implementations, and drastically lowers the risk of non-converging executions, especially to higher precision.

2. Background and related work

Adaptive parallel SGD. (cf. also Table 1) Staleness/asynchrony awareness was first studied for smooth and convex problems in (Agarwal & Duchi, 2011), introducing a step size reduction based on *worst-case* staleness. Adaptiveness to *observed* staleness was studied in (McMahan & Streeter, 2014) assuming convexity, sparse gradients and certain ordering of *reads* and *updates* across threads, and evaluated for logistic regression. (Zhang et al., 2016), with a $1/\tau$ staleness compensation scheme in a *semi-synchronous* distributed settings, show empirically speedup for ANN training with limited parallelism. For *partial asynchrony*, (Haddadpour et al., 2019) introduced an adaptive scheme for regulating synchronization frequency, show-

Table 1. Staleness-adaptive *AsyncSGD*: literature highlights and new contributions.

	Online	Strategy	Mean-pres.	Prio-pres.	Convergence guarantees	Evaluation
(Agarwal & Duchi, 2011)	×	-	×	-	Convex	LR[1]
(McMahan & Streeter, 2014)	✓	-	×	-	Convex	LR[1]
(Zhang et al., 2016)	✓	$1/\tau$ (static)	×	✓	Non-convex	ANN
AdaDelay (Sra et al., 2016), (Aviv et al., 2021)	✓	$\mathcal{O}(1/\sqrt{\tau})$ (static)	×	✓	Convex	LR[1]
MindTheStep-AsyncSGD (Bäckström et al., 2019)	✓	Model-based (static)	✓	-	Convex	ANN
(Ren et al., 2020)	✓	$\mathcal{O}(t^{-\tau})$ (static)	×	✓	Convex+[2]	LR+[2]
FLeet (Damaskinos et al., 2020)	✓	$\mathcal{O}(e^{-\beta\tau})$ (semi-dynamic)	×	✓	-	ANN
ASAP.SGD	✓	Dynamic	✓	✓	Non-convex, PL	ANN

[1] Logistic regression [2] includes star- and quasi convex, and the Rosenbrock test function.

ing convergence bounds in non-convex Polyak-Lojasiewicz functions. In contrast, our work establishes convergence bounds for *fully asynchronous* SGD with unbounded staleness, using staleness-adaptive step size strategies, for general non-convex functions, as well as ones satisfying the Polyak-Lojasiewicz condition.

AdaDelay (Sra et al., 2016) proposed $\mathcal{O}(1/\sqrt{\tau})$ staleness-adaptive step sizes for smooth, convex problems, showing scalability improvements. Their analysis was based on a uniformly distributed staleness model, which (Bäckström et al., 2019) established to be a simplifying assumption; the latter also introduced the *MindTheStep-AsyncSGD* framework, proposing $\mathcal{O}(C^{-\tau})$ and $\mathcal{O}(C^{-\tau}/\tau!)$ schemes based on a Poisson-based staleness model, showing improved convergence rates for practical DL. (Aviv et al., 2021) introduced a regret-based delay-adaptive approach for convex, smooth settings, while in a Federated Learning (FL) context, (Damaskinos et al., 2020) adopted an exponential dampening approach ($\mathcal{O}(C^{-\beta\tau})$), explored initially in (Bäckström et al., 2019), where the rate of decay is based on the s-th percentile of the staleness distribution. The approach of (Damaskinos et al., 2020) showed practical benefits for online ML applications at the edge. In (Ren et al., 2020), a $\mathcal{O}(t^{-\tau})$ staleness-adaptive scheme is proposed, analyzed under quasi- and star-convex functions, showing improved convergence for projected GD with artificial noise, under simulated staleness. The aforementioned works are however mostly *static* in their strategy, i.e. are either model-based or based on a heuristic (e.g. $1/\tau$).Here we study closely the staleness distribution and explore how to utilize this information to *generate the strategy itself*, which (see Section 6) entails significant improvements in convergence and robustness.

Non-convex asynchronous SGD. The literature on standard, non-adaptive, convex *AsyncSGD* is vast, and a useful overview is in (Ben-Nun & Hoefler, 2019). Here we highlight recent relevant *AsyncSGD* results for practical non-convex applications, including DL. In (Yazdani & Hale, 2021), linear convergence was established under the Polyak-Lojasiewicz condition, however assuming bounded staleness; the empirical evaluation focused on logistic regression, a convex problem. (Wei et al., 2019) proposed a static method for ensuring data-disjoint concurrent accesses, showing promising scalability for the non-convex problem of matrix factorization with SGD; the method is however, as they state, not applicable to DL in general. (Lopez et al., 2020) proposed a semi-asynchronous SGD approach, showing speedup for DL on CPU and GPU architectures, requiring a synchronizing master thread which averages updates (Xie et al., 2020) proposed a byzantine-tolerant asynchronous SGD framework, using a parameter server ensuring quality and relevance of updates. Similarly to ours, their work covers Polyak-Lojasiewicz problems, however to the best of our knowledge, our work is the first to do so for instance-based asynchrony-aware algorithmic implementations of *AsyncSGD*.

Optimization problem. We consider the unconstrained optimization problem

$$\underset{\theta\in\mathbb{R}^d}{\text{minimize}} \quad L_D(\theta) \tag{1}$$

where (i) D is the data set to be processed, (ii) $\theta \in \mathbb{R}^d$ is the ML model that encodes the learned knowledge of D and (iii) the target function $L\colon \mathbb{R}^d \to \mathbb{R}^+$ quantifies the loss (error) of θ over D. Given some randomly chosen initial θ_0, the first-order iterative optimizer SGD repeats the following:

$$\theta_{t+1} = \theta_t - \eta_t \nabla\tilde{L}(\theta_t) \tag{2}$$

where $\theta_t \in \mathbb{R}^d$ is the state of the model θ, and η_t is the step size, in iteration t. We assume that $\tilde{L} = L_B$ where $B \subset D$ is a uniformly sampled mini-batch of data, and that $\tilde{L}$ is an unbiased estimator of L_D, i.e. $\mathbf{E}\left[\tilde{L}(\theta)\right] = L_D(\theta)\ \forall\theta \in \mathbb{R}^d$. We assume that mini-batch samples, and hence the stochastic gradients $\nabla\tilde{L}$, are mutually statistically independent. The loss function $L_D\colon \mathbb{R}^d \to \mathbb{R}^+$, $\theta \mapsto L_D(\theta)$ quantifies the performance of an ANN model, parameterized by θ. SGD is repeated until θ satisfies ϵ-convergence, defined as $\|L(\theta) - L(\theta^*)\| < \epsilon$, θ^* being a global minimum of L.

3. System model and problem analysis

We consider a system with m concurrent asynchronous threads or processes. Threads follow the SGD rule of (2) asynchronously, within the boundaries of the algorithm which implements it, being responsible for potential guarantees on consistency and progress. An outline of a standard shared-memory parallel *AsyncSGD* implementation is pro-

Algorithm 1 Staleness-adaptive shared-memory *AsyncSGD*

GLOBAL loss function L, iteration counter t, max. n.o. iterations T, shared state θ, step size function $\eta(\tau)$

```
1: Let $(t, \theta) \leftarrow (0, \text{rand\_init}())$          Randomly initialize $\theta$
2: Each thread:
3: while $t < T$ do
4:     $(t_{local}, \theta_{local}) \leftarrow \text{copy}(t, \theta)$
5:     $\nabla_{local} \leftarrow \nabla\tilde{L}(\theta_{local})$          Compute local random gradient
6:     $(t', \theta') \leftarrow \text{copy}(t, \theta)$          Acquire latest state
7:     $\tau \leftarrow t' - t$          Calculate staleness
8:     $(t, \theta) \leftarrow (t' + 1, \theta' - \eta(\tau)\nabla_{local})$
9: end while
```

vided in Algorithm 1, showing also how a staleness-adaptive step size is introduced. In direct shared memory communication, threads have atomic access to single-word locations for read and modify operations; in arbitrary contexts, steps 4 and 8 in Algorithm 1 can be executed through the help of a parameter server, through requests to read and update θ. Due to asynchronous reads and updates of θ there can be intermediate updates, hence, the progression of θ follows:

$$\theta_{t+1} = \theta_t - \eta_t \nabla\tilde{L}(v_t) \quad (3)$$

where $v_t = \theta_{t-\tau_t}$ is the view of the updating thread in iteration t, and τ_t is the staleness, defined as the number of *intermediate* updates. When atomicity is not guaranteed, e.g. HOGWILD!, a total ordering of the updates is not naturally imposed, and has to be defined. Here, we assume a total ordering as in (Alistarh et al., 2018), and define the staleness thereafter. By *staleness distribution* we refer to the distribution of all observed staleness values throughout a particular execution of *AsyncSGD*. We consider staleness (τ_t) to be a stochastic process, the elements of which, unlike the stochastic gradients, are not necessarily mutually independent. However, we assume $\mathbf{E}[\tau_t] = \bar{\tau}\ \forall t$ and that the execution is *non-anticipative* in the sense that states are mean-independent of future instances of the staleness, in particular

$$\mathbf{E}[\tau_t \mid \tau_{t'}] = \mathbf{E}[\tau_t]\ \forall t < t' \quad (4)$$

since the expected staleness is not influenced by future staleness. Note that this implies $\mathbf{E}[\tau_t \tau_{t'}] = \mathbf{E}[\tau_t]\mathbf{E}[\tau_{t'}]$. Similarly, we assume that the staleness is mean-independent of θ, since the statistical properties of the convergence progress are not expected to influence the delays of individual threads, which are related to hardware and scheduling.

The step size η_t in iteration t will, unless stated otherwise, be considered a function of the staleness τ_t in the same iteration; the details appear in the subsequent section. In the analysis section, we will generally use the notation $\mathbf{E}[x] = \bar{x}$.

Dampening is not sufficient. The primary focus of previous approaches has been to dampen the step size of stale updates (cf. Table 1). However, the overall staleness distribution changes with higher parallelism (Sra et al., 2016; Bäckström et al., 2021), in particular for m threads, $\mathbf{E}[\tau] \approx m - 1$ is known to hold (Bäckström et al., 2019). An adaptive step size which merely diminishes stale updates, will consequently tend to very small values as m increases. This introduces a scalability issue, especially for non-convex problems, such as DL, which are step size sensitive, requiring updates of sufficiently coarse granularity to retain the level of stochasticity necessary for convergence. In fact, the commonly adopted $\mathcal{O}(\tau^{-1})$ scaling, as well as the FLeet $\mathcal{O}(e^{-\beta\tau})$ exponential dampening, were evaluated in our study, and compared with both constant step size *AsyncSGD*, as well as the staleness-adaptive function TAIL-τ proposed here. The results show convergence rates *orders of magnitude* slower than just the corresponding non-adaptive variant, and especially when compared to the proposed TAIL-τ function (see Section 6).

4. Method

Here we present the ASAP.SGD framework, the staleness-adaptive TAIL-τ step size function, and their properties.

4.1. The ASAP.SGD framework

We start by formalizing the concept of a staleness-adaptive step size, and its connection to the overall base step size.

Definition 4.1. A staleness-adaptive step size function $\eta\colon \mathbb{N} \to \mathbb{R}^+$, $\tau \mapsto \eta(\tau : \eta^0)$, given some base step size $\eta^0 \in \mathbb{R}^+$, maps the stochastic staleness τ_t of the update at time t onto a step size $\eta(\tau : \eta^0)$ to be used for that update.

The base step size η_0 is typically the 'best known'step size for the problem at hand for standard sequential SGD. A staleness-adaptive function then alters this step size online, based on observed staleness. Definition 4.1 implies in particular that the step size, as a function of the staleness, is consequently also considered stochastic. The step size η_t at iteration t is however influenced only by the staleness τ_t.

A challenge with staleness-adaptive step sizes is that they may alter the overall magnitude of the updates, which is undesirable since (i) it induces deviation from the expected step size magnitude, with unpredictable impact on the convergence, and (ii) makes it inherently different to compare the convergence impact of different strategies. We introduce the *mean-preservation* property to address this.

Definition 4.2 (Mean-preservation)**.** A staleness-adaptive step size function η is referred to as *mean-preserving* if

$$\mathbf{E}\big[\eta(\tau : \eta^0)\big] = \eta^0 \quad (5)$$

Definition 4.2 ensures that the average step size used throughout an execution of *AsyncSGD* is exactly η^0. Performance benefits due to a *mean-preserving* adaptive step size can hence be assured to be due to the *adaptation* strategy, as opposed to using a step size of an overall different magnitude. In the following, in the context of mean-preserving step sizes, we use notation η^0 and $\bar{\eta}$ interchangeably.

Lastly, we introduce the *priority-preservation* property:

Definition 4.3 (Priority-preservation)**.** A staleness-adaptive step size function η is referred to as *priority-preserving* if η is *non-increasing* with respect to τ, i.e.

$$\eta(\tau+1:\eta^0) \leq \eta(\tau:\eta^0) \;\; \forall\tau \tag{6}$$

Definition 4.3 implies not only that *stale* updates are lower prioritized, but also that *fresh* updates can be *emphasized*. With the above in mind, we now define the ASAP.SGD framework for staleness-adaptive step size functions:

Definition 4.4 (ASAP.SGD)**.** A function η is an ASAP.SGD step size function, iff η is *staleness-adaptive*, *mean-preserving* and *priority-preserving*.

4.2. The TAIL-τ function

Next, within the ASAP.SGD framework we define the instance-based staleness-adaptive TAIL-τ step size function, which considers the overall *staleness distribution*, to dynamically compute an execution-tailored adaptation strategy.

Definition 4.5. A TAIL-τ function is a staleness-adaptive step size function η which is of the form

$$\eta(\tau:\eta^0) = C_A(\tau)\cdot\eta^0 \tag{7}$$

where the scaling factor C_A is given by

$$C_A(\tau) = 1 + A\cdot(1-2F_{\bar{\tau}}(\tau))$$

Here, $F_{\bar{\tau}}(\tau) = \mathbf{P}[\bar{\tau}\leq\tau]$ is the cumulative distribution function (CDF) of the stochastic staleness.

The amplitude parameter A of the TAIL-τ function specifies the maximum deviation of η from the base step size η^0. The scaling factor $C_A(\tau)$ of the TAIL-τ function (7) utilizes the CDF of τ, implicitly taking the system staleness distribution into consideration for generating a dynamic adaptive step size function, tailored to the specific execution. This is visualized in Figure 1, showing the response of $C_A(\tau)$ to different ranges of stochastic staleness. E.g., it enables tailored treatment of *multi-modal staleness distributions*, which may emerge under hyperthreading or congestion for accessing shared resources. The name TAIL-τ relates to the $1-F_{\bar{\tau}}(\tau) = \mathbf{P}[\bar{\tau}>\tau]$ component, which is known as the *tail distribution* function.

Practical note. The TAIL-τ function is straight-forward to apply to any implementation of *AsyncSGD*, since it requires only measuring the distribution of the staleness, and computing the corresponding CDF $F_{\bar{\tau}}(\tau) = \mathbf{P}[\bar{\tau}<\tau]$, to be used in the step size (7). TAIL-τ introduces negligible overhead, as measuring τ is a small, constant-time operation, independently of e.g. architecture size.

Next, we verify that TAIL-τ satisfies the properties of the ASAP.SGD framework. Proofs appear in the Appendix.

Theorem 4.6. *A* TAIL-τ *function* η *according to (7) is an* ASAP.SGD *step size function, according to Definition 4.4*

In the following lemma we show several core properties of the TAIL-τ function, to be used in subsequent analysis.

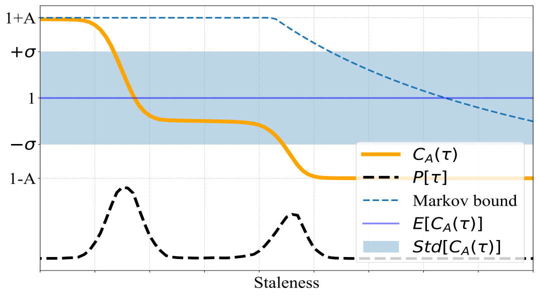

Figure 1. The scaling factor $C_A(\tau)$ of TAIL-τ*, relative a staleness distribution, with properties from Lemma 4.7. Fresh updates (low staleness) are emphasized, stragglers are dampened.*

Lemma 4.7. *For a* TAIL-τ *adaptive step size* $\eta(\tau:\eta^0) = C_A(\tau)\cdot\eta^0$, *we have*

$$\max_\tau \eta(\tau:\eta^0) = (1+A)\eta^0 \tag{i}$$

$$\min_\tau \eta(\tau:\eta^0) = (1-A)\eta^0 \tag{ii}$$

$$Var[\eta(\tau)] = (A\bar{\eta})^2/3 \tag{iii}$$

$$C_A(\tau) \leq 1 + (2(m-1)/\tau - 1)\,A \tag{iv}$$

Lemma 4.7 provides useful criteria for deciding the parameters η^0 and A in practice, for ensuring that $\eta(\tau:\eta^0)$ stays within desirable magnitudes suitable for the given problem.

TAIL-τ can be extended in the same spirit as Definition 4.5, to allow a *higher degree of customizability* in the instance-adaptiveness, still satisfying the properties of Lemma 4.7. In particular, the degree with which the CDF of τ influences the scaling factor C_A, and how much variations of τ values are reflected in the step size, can be altered, depending on the application. The general formulation of TAIL-τ enjoys the convergence guarantees to follow in Section 5, however, appears in Appendix A.1 due to space constraints.

5. Convergence analysis

In this section we establish asymptotic convergence bounds of the method proposed in the previous section, considering:

Assumption 5.1. Expected Lipschitz-continuous gradients

$$\mathbf{E}[\|\nabla L(x) - \nabla L(y)\|] \leq \mathcal{L}\mathbf{E}[\|x-y\|] \;\forall x,y \tag{8}$$

Assumption 5.2. Expected bounded gradient moment

$$\mathbf{E}\left[\|\nabla L(x)\|^2\right] \leq M^2 \;\forall x \tag{9}$$

These provide the problem additional structure, hold for a wide set of loss functions in practice, and are widely adopted in the literature (Agarwal & Duchi, 2011; Zhang et al., 2016; Alistarh et al., 2018; Bäckström et al., 2019). We consider general non-convex problems here, but for self-containment we also establish fundamental convex convergence bounds, available in the Appendix.

The following lemma shows the expected progression of the SGD iterates for arbitrary staleness-adaptive step sizes.

Lemma 5.3. *Consider the optimization problem of (1), and follow the SGD step (3). Let* $\eta_t = \eta(\tau_t;\bar{\eta})$ *be a staleness-*

adaptive step size function (according to Definition 4.1). Then we have the following expected iterative progression:

$$\mathbf{E}[L(\theta_{t+1}) - L(\theta_t)]$$
$$\leq \mathcal{L}M^2\left(\mathbf{E}\left[\eta^2\right]/2 + \mathbf{E}[\tau\eta]\mathbf{E}[\eta]\right) - \mathbf{E}[\eta]\mathbf{E}\left[\|\nabla L(\theta_t)\|^2\right]$$

The following lemma establishes the expected iterative progression of *AsyncSGD* with ASAP.SGD step sizes:

Corollary 5.4. *Under the same conditions as Lemma 5.3, let in addition $\eta(\tau_t;\bar\eta)$ be mean- and priority-preserving, hence an* ASAP.SGD *step size. Then we have:*

$$\mathbf{E}[L(\theta_{t+1}) - L(\theta_t)] \leq \mathcal{L}M^2\left(\mathbf{E}\left[\eta^2\right]/2 + \bar\eta^2\bar\tau\right) - \bar\eta\mathbf{E}\left[\|\nabla L(\theta_t)\|^2\right]$$

Corollary 5.4 follows directly from mean-preservation, and from that $\mathbf{E}[\tau\eta] \leq \mathbf{E}[\tau]\mathbf{E}[\eta]$ by priority-preservation. Corollary 5.4 serves as a common starting point for convergence analysis of a wide range of ASAP.SGD step size functions. Using the specifics of the step size function, lemma 5.3 can be used to derive explicit convergence bounds, as we demonstrate in the following for TAIL-τ.

Corollary 5.5. *Assume the conditions of Lemma 5.3, and let $\eta(\tau_t;\bar\eta) = C_A(\tau)\cdot\bar\eta$ be a* TAIL-τ *step size function. Then:*

$$\mathbf{E}[L(\theta_{t+1}) - L(\theta_t)] \leq \mathcal{L}M^2\bar\eta^2\left(A^2/6 + \bar\tau\right) - \bar\eta\mathbf{E}\left[\|\nabla L(\theta_t)\|^2\right]$$

Corollary 5.5 follows from Corollary 5.4, utilizing the TAIL-τ properties from Lemma 4.7, and shows the iterative improvement for TAIL-τ, to be used in subsequent results.

In the following theorem we establish expected time until convergence to an approximate critical point of SGD for general non-convex, smooth functions, using TAIL-τ.

Theorem 5.6. *Assume $L(\theta_0) - L(\theta^*) < \delta$, and let $\eta(\tau_t;\bar\eta) = C_A(\tau)\cdot\bar\eta$ be a* TAIL-τ *step size function. Then we have a $\mathcal{O}(1/\sqrt{T})$ convergence bound, to an approximate critical point, after T* AsyncSGD *iterations. Specifically, if*

$$\bar\eta = \sqrt{\delta/\mathcal{L}M^2T(A^2+\bar\tau)} \tag{10}$$

then $\min_t \mathbf{E}\left[\|\nabla L(\theta_t)\|^2\right] \leq 2\sqrt{\mathcal{L}M^2(A^2+\bar\tau)\delta/T}$

Theorem 5.6 shows that TAIL-τ recovers the standard bound $\mathcal{O}(1/\sqrt{T})$ of SGD on smooth non-convex problems (Ghadimi & Lan, 2013). However, this is rather pessimistic as it applies to general non-convex functions. We consider the following to provide more structure to the problem:

Assumption 5.7. Polyak-Lojasiewicz (PL) condition. A function L is referred to as μ-PL if, for some $\mu > 0$:

$$\mathbf{E}\left[\|\nabla L(x)\|^2\right] \geq \mu\,\mathbf{E}[L(x) - L(x^*)]\ \forall x \tag{11}$$

PL is a geometric condition characterizing the shape of non-convex functions. It can be regarded as a generalization of strong convexity, however without requirements on e.g. uniqueness of minimizers. Several ML loss functions satisfy the condition, including least squares, logistic regression, support vector machines (Karimi et al., 2016) and certain types of deep ANNs (Charles & Papailiopoulos, 2018).

Next we establish asymptotic convergence of *AsyncSGD* with TAIL-τ step size function for smooth, μ-PL functions.

Theorem 5.8. *Let L be μ-PL, and $L(\theta_0) - L(\theta^*) < \delta$. Further, let $\eta(\tau_t;\bar\eta) = C_A(\tau)\cdot\bar\eta$ be a* TAIL-τ *step size function. Then we have expected ϵ-convergence in $T = \mathcal{O}\left(\frac{1}{\epsilon}\log\left(\frac{2\delta}{\epsilon}\right)\right)$ iterations. More precisely, let*

$$\bar\eta = \frac{\mu\epsilon}{\mathcal{L}M^2(A^2/3+2\bar\tau)} \tag{12}$$

Then we have $\mathbf{E}[L(\theta_T) - L(\theta^*)] < \epsilon$ for

$$T > \frac{\mathcal{L}M^2(A^2/3+2\bar\tau)}{\mu^2\epsilon}\log\left(\frac{2\delta}{\epsilon}\right)$$

The reason for the $O(A^2)$ term in Theorems 5.6 and 5.8 (negligible compared to the $O(\tau)$ additive term) is due to the analytical estimation of the variance of TAIL-τ (Lemma 4.7, (iii)), which must be considered (Cor. 5.5), specifically when expanding the $E[\eta^2]$ term. As we make no assumptions on PDF(τ), the bounds reflect the expected worst-case convergence over all possible τ distributions. Additional information on the τ distribution can (using Corollary 5.4 as a starting point) derive tighter algorithm-specific bounds.

6. Evaluation

We complement the analysis with benchmarking the TAIL-τ function, for implementations of *AsyncSGD* representative of a variety of system-execution properties relating with scheduling and ordering. We evaluate TAIL-τ, comparing to standard constant step size executions, for relevant DL benchmark applications, namely training the LeNet (LeCun et al., 1998) architecture, as well as a 3-layer MLP, for image recognition for image recognition on both MNIST and Fashion MNIST. The evaluation focuses on convergence rates, primarily wall-clock time to ϵ-convergence (which is the most relevant in practice), as well as number of successful executions, for various precision levels ϵ. We include a broad spectrum of parallelism, giving a detailed picture of the capability of the methods to scale in practice. We also study the staleness distributions, the adaptive response of the TAIL-τ function, and its impact on the convergence.

Implementation. We evaluate the TAIL-τ for three *AsyncSGD* algorithms, representing fundamentally different synchronization mechanisms and guarantees on progress and consistency, and in this way capturing a variety of scheduling and ordering system-execution properties: (i) lock-based *AsyncSGD* (Zinkevich et al., 2009; 2010; Agarwal & Duchi, 2011) (ii) lock-free, but inconsistent, HOGWILD! (Recht et al., 2011; De Sa et al., 2015; Alistarh et al., 2018) and (iii) the lock-free and consistent *Leashed-SGD* algorithm (Bäckström et al., 2021), denoted ASYNC, HOG and LSH, respectively. Executions using our TAIL-τ function are indicated by the suffix _TAIL. The implementation extends the open *Shared-Memory-SGD* (Bäckström, 2021) C++ library, connecting ANN operations to low-level implementations of parallel SGD, and is free to use for research purposes.

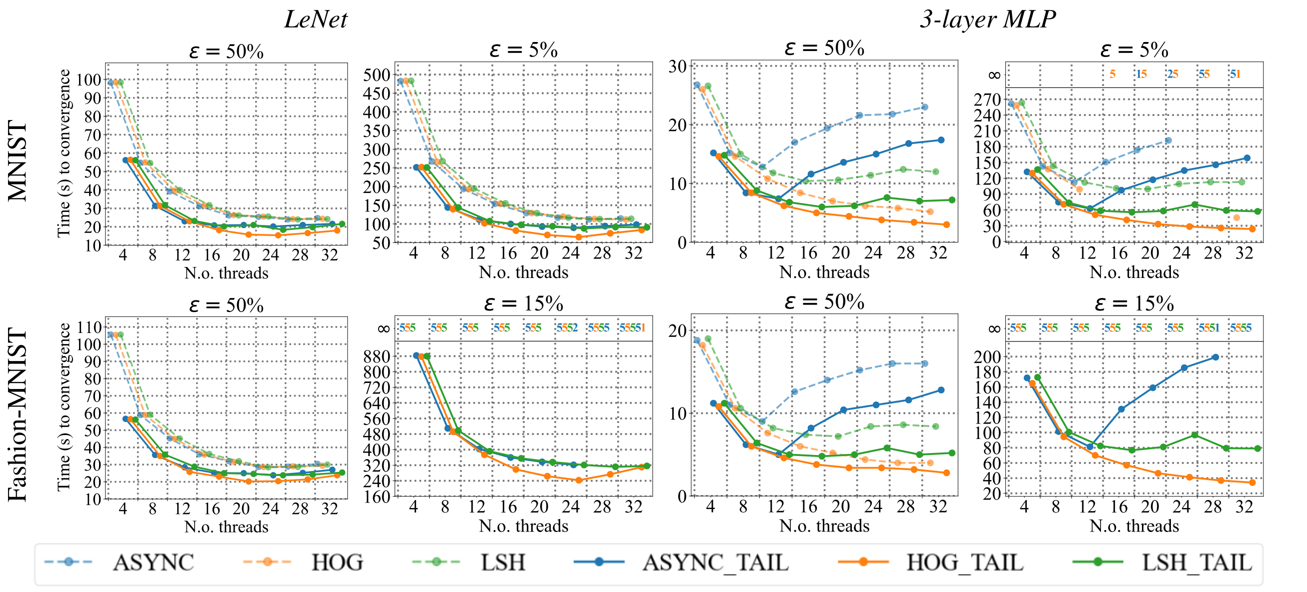

Figure 2. Convergence rates for LeNet (*left*) and a 3-layer MLP (*right*) for MNIST and Fashion-MNIST recognition training with *AsyncSGD*, with HOGWILD! (HOG), *Leashed-SGD* (LSH), and traditional lock-based (ASYNC) implementations. Executions using the instance-based TAIL-τ staleness-adaptive step size are indicated with the suffix _TAIL.

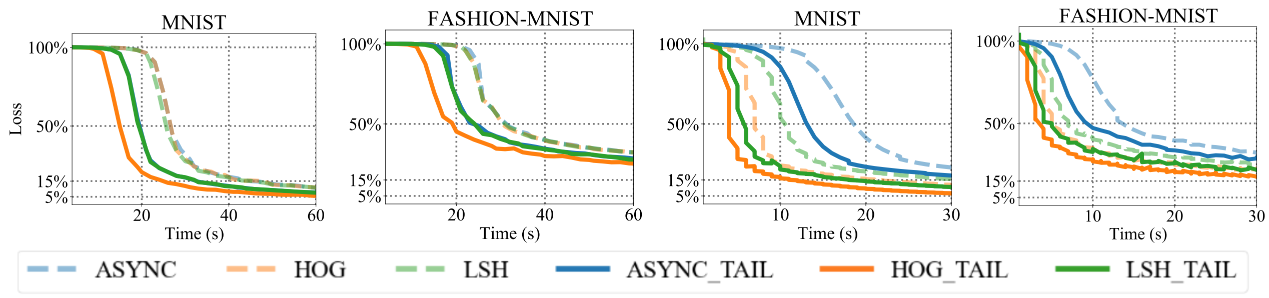

Figure 3. Loss over time for HOGWILD! (HOG), *Leashed-SGD* (LSH), and traditional lock-based (ASYNC) *AsyncSGD* with parallelism $m = 20$, with and without the TAIL-τ staleness-adaptive step size (suffix _TAIL) for LeNet (*left*) and a 3-layer MLP (*right*).

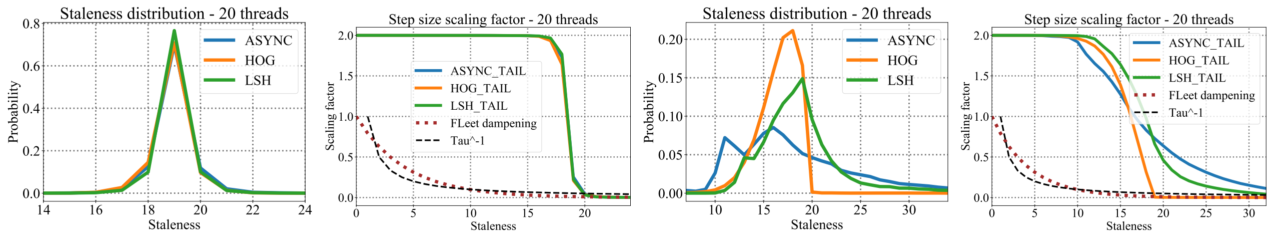

Figure 4. Staleness distribution, and the TAIL-τ scaling factor for LeNet (*left*) and a 3-layer MLP (*right*), for MNIST and Fashion-MNIST.

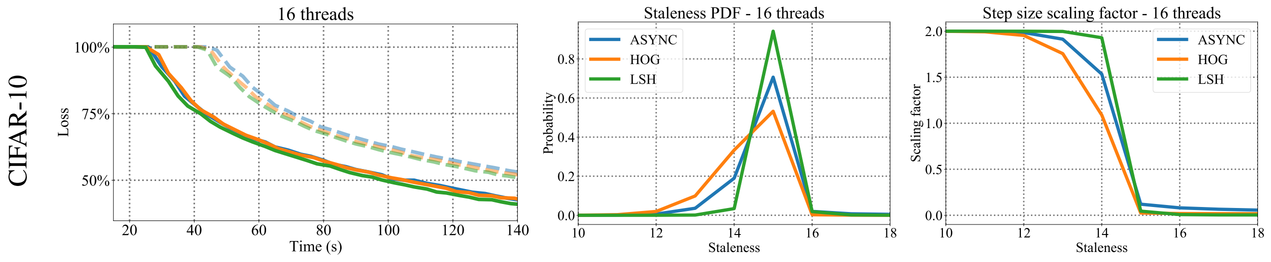

Figure 5. Loss over time (*left*), staleness distribution (*center*), and the TAIL-τ scaling factor (*right*) for LeNet training on CIFAR-10.

Table 2. Results overview - speedup across the parallelism spectrum achieved by the TAIL-τ step size, relative a standard constant one.

Speedup		*50%-convergence*				*5% (15%)-convergence*			
Dataset	*Architecture*	min	max	avg	success*	min	max	avg	success*
MNIST	LeNet	1.12	1.75	1.51	1.0	1.16	1.92	1.6	1.0
	MLP	1.30	2.0	1.66	1.0	1.42	1.99	1.82	3.53
Fashion-MNIST	LeNet	1.13	1.90	1.48	1.0	∞	∞	∞	∞
	MLP	1.25	1.8	1.56	1.0	∞	∞	∞	∞
CIFAR-10	LeNet	1.03	1.45	1.29	1.0	-	-	-	-

*Ratio between n.o. executions that reached the desired precision for TAIL-τ executions vs. standard *AsyncSGD*

Experiment setup. We tackle the problem of ANN training for image classification on the datasets MNIST (LeCun & Cortes, 2010) of hand-written digits, CIFAR-10 (Krizhevsky et al., 2009) of everyday objects, and Fashion-MNIST (Xiao et al., 2017) of clothing article images. All datasets contain 60k images, each belonging to one of ten classes $\in \{0, \ldots, 9\}$. For this, we train a LeNet CNN architecture, as well as a 3-layer MLP, with 128 neurons per layer (denoted MLP in the following), for 100 epochs. We use standard settings and hyper-parameters; For MNIST and Fashion-MNIST training we use a base step size of $\eta_0 = 1\text{e}{-4}$ and mini-batch size 128, while for CIFAR-10 we use $\eta_0 = 5\text{e}{-3}$ and a mini-batch size of 16. The *multi-class cross-entropy* loss function is used in all experiments. For *Leashed-SGD*, we use the default setting of an infinite persistence bound. We use a TAIL-τ step size function (as in Definition 4.5), that adapts to each unique execution, based on the measured staleness distribution, with an adaptation amplitude of $A = 1$, due to its role in emphasizing fresh updates and dampening stragglers. The experiments are conducted on a 2.10 GHz Intel(R) Xeon(R) E5-2695 two-socket 36-core (18 cores per socket, each supporting two hyper-threads), 64GB non-uniform memory access (NUMA), Ubuntu 16.04 system.

Plots show averaged values from 5 executions for each setting, unless otherwise stated. ϵ-convergence is achieved when $L(\theta) < \epsilon$, expressed as % of the initial loss $L(\theta_0)$. The number of executions that fail to reach ϵ-convergence is indicated by ∞ at the top, if such executions occurred. This is important, since such executions result in models of insufficient accuracy, and thereby are wasted work.

Convergence speedup and scalability. We measure convergence time to first 50% of the initial error (i.e. 50%-convergence), and then to higher precision (5% for MNIST, and 15% for Fashion-MNIST to enable clearer comparison, since baselines rarely converge to this level of precision) across the parallelism spectrum. The results (Figure 2) show that the TAIL-τ step size function yields persistent and substantial speedup in convergence time (12% in the worst-case, 100% in the best), for all combinations of datasets, architectures and *AsyncSGD* implementations (see Table 2 for details). Training plots, showing loss progression over time, are shown in Figure 3, demonstrating the convergence speed being *orders of magnitudes* faster than standard *AsyncSGD*. Similar speedup is observed for training on the CIFAR-10 dataset, shown in Figure 5.

For higher-precision convergence, non-converging executions are frequent among the standard *AsyncSGD* algorithms, especially under higher parallelism. Their ability to converge varies, demonstrating the impact of underlying synchronization and progress guarantees. We observe in particular that HOGWILD! achieves fastest overall convergence, while *Leashed-SGD* provides higher reliability, i.e. lower risk of non-convergence. However, independently of the properties of the *AsyncSGD* implementation, we observe, in addition to persistently faster convergence, also that the TAIL-τ step size ensures a significantly lower risk of non-convergence, hence higher reliability (see Table 2).

Staleness-based dampening according to the commonly adopted $\times\tau^{-1}$ scheme, as well as the FLeet [1] exponential dampening (Damaskinos et al., 2020), are evaluated for MLP and LeNet training on MNIST, compared to standard *AsyncSGD* with constant step size. The results (Figure 6) show, as conjectured in Section 3, convergence of significantly slower rate compared to a constant step size, which in addition (as opposed to TAIL-τ, as well as constant step size) decays with higher parallelism due to the reduced overall step size magnitude. The speedup of TAIL-τ compared to constant step size is shown in Figure 2 and 3.

Instance-based adaptiveness. Figure 4 shows the staleness distribution of the *AsyncSGD* algorithms for LeNet and MLP, along with the corresponding TAIL-τ step size scaling factors, generated dynamically based on the staleness distribution of the particular execution. We observe, as expected from Section 4, that the staleness-adaptive step size strategies, generated by TAIL-τ, emphasize fresh updates and diminishe the impact of stale ones, taking into consideration the underlying execution-specific staleness distribution. This, as shown above, results in increased stability, i.e. lowered risk of non-convergence, as well as significant increase in convergence rates. Note that considering standard SGD eliminates side-effects of 'add-ons', enabling clearer comparisons. Other step-size-altering methods (e.g. ADAM or schedules) can be used in conjunction, by applying them to

[1]As in the original paper, we use the dampening factor $\Lambda(\tau) = e^{-\beta\tau}$, where β satisfies $(\tau_{thres}/2 + 1)^{-1} = e^{-\beta\tau_{thres}/2}$, where τ_{thres} is the beginning of the staleness distribution *tail*, which we consider to be $\tau_{thres} = m + 1$ here, based on observation. The similarity-based boosting option was not considered here.

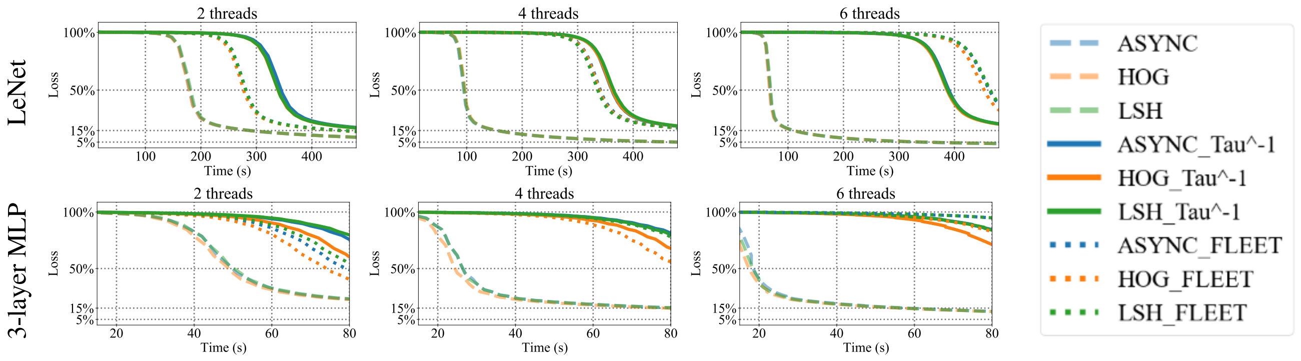

Figure 6. AsyncSGD training on MNIST with HOGWILD! (HOG), *Leashed-SGD* (LSH), and traditional lock-based (ASYNC) implementations, comparing executions using the FLeet exponential dampening approach (suffix: _FLEET) and the $\times\tau^{-1}$ staleness-adaptive scheme (suffix: _Tau$^-$1) against standard, constant step size.

η_0 (Def. 4.4) and is a relevant target of future studies.

System-related insights. The emerging τ distribution of an AsyncSGD execution is influenced by many underlying factors, including (i) the compute infrastructure (UMA/NUMA, hyper-threading), (ii) consistency guarantees, and the associated synchronization mechanisms (e.g. loose consistency as in HOGWILD!, lock-based, or *Leashed-SGD* etc), (iii) gradient computation vs application time, (iv) number of threads. Several works (Ben-Nun & Hoefler, 2019; Bäckström, 2021) are dedicated to studying this, however the nature of this dependency is an open problem. Although this is not the main scope here, we see that implicitly adapting to the aforementioned aspects, through the PDF(τ) signature, yields significant practical benefits, as we show with TAIL-τ.

7. Conclusion

We introduce ASAP.SGD - a framework for capturing essential properties of general staleness-adaptive step size functions for *AsyncSGD*, providing structure to the domain of staleness-adaptiveness, and can guide the design of new adaptive step size strategies. Within ASAP.SGD, we introduce the first instance-based dynamic step size function, TAIL-τ, which generates a tailored adaptiveness strategy for each unique execution. We analyze general ASAP.SGD functions for *AsyncSGD*, as well as TAIL-τ in particular, and recover convergence bounds for both convex and non-convex problems, as well as establish new bounds for ones satisfying the Polyak-Lojasiewicz condition.

We implement TAIL-τ, extending existing *AsyncSGD* implementations, to provide a platform for further research in the domain. The evaluation covers three implementations of *AsyncSGD*, with fundamentally different algorithmic properties, for training LeNet and an MLP for image recognition on MNIST and Fashion-MNIST. The results show that TAIL-τ is a vital component for *AsyncSGD* practical deployments, due to its ability to, based on the properties of the unique execution, generate an adaptiveness strategy tailored to the specific execution, yielding persistent speedup across the entire parallelism spectrum, and tremendous increase in reliability to converge, in particular to high precision. Efficiency is important not only for timeliness, but also for resource utilization, considering especially how energy-consuming is e.g. ANN training (Sorbaro et al., 2020).

Acknowledgements

This work is supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP), Knut and Alice Wallenberg Foundation; the Swedish Research Council (Vetenskapsrådet) – projects: "EPITOME" 2021-05424 and "Relaxed Concurrent Data Structure Semantics for Scalable Data Processing" 2021-05443; and Chalmers AoA frameworks Energy and Production, proj. INDEED, and WP "Scalability, Big Data and AI".

References

Agarwal, A. and Duchi, J. C. Distributed delayed stochastic optimization. In *Advances in Neural Information Processing Systems*, pp. 873–881, 2011.

Alistarh, D., De Sa, C., and Konstantinov, N. The convergence of stochastic gradient descent in asynchronous shared memory. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, pp. 169–178, 2018.

Aviv, R. Z., Hakimi, I., Schuster, A., and Levy, K. Y. Asynchronous distributed learning: Adapting to gradient delays without prior knowledge. In *International Conference on Machine Learning*, pp. 436–445. PMLR, 2021.

Bäckström, K., Papatriantafilou, M., and Tsigas, P. Mindthestep-asyncpsgd: Adaptive asynchronous parallel stochastic gradient descent. In *2019 IEEE International*

Conference on Big Data (Big Data), pp. 16–25. IEEE, 2019.

Bäckström, K., Walulya, I., Papatriantafilou, M., and Tsigas, P. Consistent lock-free parallel stochastic gradient descent for fast and stable convergence. In *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 423–432. IEEE, 2021.

Ben-Nun, T. and Hoefler, T. Demystifying parallel and distributed deep learning: An in-depth concurrency analysis. *ACM Computing Surveys (CSUR)*, 52(4):1–43, 2019.

Bäckström, K. shared-memory-sgd. https://github.com/dcs-chalmers/shared-memory-sgd, 2021.

Charles, Z. and Papailiopoulos, D. Stability and generalization of learning algorithms that converge to global optima. In *International Conference on Machine Learning*, pp. 745–754. PMLR, 2018.

Damaskinos, G., Guerraoui, R., Kermarrec, A.-M., Nitu, V., Patra, R., and Taiani, F. Fleet: Online federated learning via staleness awareness and performance prediction. In *Proceedings of the 21st International Middleware Conference*, pp. 163–177, 2020.

De Sa, C. M., Zhang, C., Olukotun, K., and Ré, C. Taming the wild: A unified analysis of hogwild-style algorithms. *Advances in neural information processing systems*, 28, 2015.

Ghadimi, S. and Lan, G. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

Haddadpour, F., Kamani, M. M., Mahdavi, M., and Cadambe, V. Local sgd with periodic averaging: Tighter analysis and adaptive synchronization. *Advances in Neural Information Processing Systems*, 32, 2019.

Karimi, H., Nutini, J., and Schmidt, M. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 795–811. Springer, 2016.

Kraska, T. Towards instance-optimized data systems, keynote. *2021 International Conference on Very Large Data Bases*, 2021.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

LeCun, Y. and Cortes, C. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lopez, F., Chow, E., Tomov, S., and Dongarra, J. Asynchronous sgd for dnn training on shared-memory parallel architectures. In *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 1–4. IEEE, 2020.

Ma, Y., Rusu, F., and Torres, M. Stochastic gradient descent on modern hardware: Multi-core cpu or gpu? synchronous or asynchronous? In *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 1063–1072. IEEE, 2019.

McMahan, B. and Streeter, M. Delay-tolerant algorithms for asynchronous distributed online learning. *Advances in Neural Information Processing Systems*, 27, 2014.

Nguyen, L., Nguyen, P. H., Dijk, M., Richtárik, P., Scheinberg, K., and Takác, M. Sgd and hogwild! convergence without the bounded gradients assumption. In *International Conference on Machine Learning*, pp. 3750–3758. PMLR, 2018.

Recht, B., Re, C., Wright, S., and Niu, F. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. *Advances in neural information processing systems*, 24, 2011.

Ren, Z., Zhou, Z., Qiu, L., Deshpande, A., and Kalagnanam, J. Delay-adaptive distributed stochastic optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 5503–5510, 2020.

Sorbaro, M., Liu, Q., Bortone, M., and Sheik, S. Optimizing the energy consumption of spiking neural networks for neuromorphic applications. *Frontiers in neuroscience*, pp. 662, 2020.

Sra, S., Yu, A. W., Li, M., and Smola, A. Adadelay: Delay adaptive distributed stochastic optimization. In *Artificial Intelligence and Statistics*, pp. 957–965. PMLR, 2016.

Wei, J., Gibson, G. A., Gibbons, P. B., and Xing, E. P. Automating dependence-aware parallelization of machine learning training on distributed shared memory. In *Proceedings of the Fourteenth EuroSys Conference 2019*, pp. 1–17, 2019.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Xie, C., Koyejo, S., and Gupta, I. Zeno++: Robust fully asynchronous sgd. In *International Conference on Machine Learning*, pp. 10495–10503. PMLR, 2020.

Yazdani, K. and Hale, M. Asynchronous parallel nonconvex optimization under the polyak-łojasiewicz condition. *IEEE Control Systems Letters*, 2021.

Zhang, W., Gupta, S., Lian, X., and Liu, J. Staleness-aware async-sgd for distributed deep learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pp. 2350–2356, 2016.

Zinkevich, M., Langford, J., and Smola, A. Slow learners are fast. *Advances in neural information processing systems*, 22, 2009.

Zinkevich, M., Weimer, M., Li, L., and Smola, A. Parallelized stochastic gradient descent. *Advances in neural information processing systems*, 23, 2010.

A. Technical Appendix

A.1. General TAIL-τ function definition

We introduce an additional degree of generalization for the TAIL-τ function, which allows significantly broader flexibility in design choices, while still satisfying the properties of the ASAP.SGD framework, and enjoying the convergence guarantees established in Section 5.

Theorem A.1. *Let a staleness-adaptive step size function η be*

$$\eta(\tau : \eta^0) = C_{s,\phi}(\tau) \cdot \eta^0$$

where the scaling factor is given by

$$C_{s,\phi}(\tau) = 1 + A \cdot (1 - 2\,\phi(F_{\tilde{\tau}}(\tau)))$$

for some amplitude factor $A \in [0,1]$ and some non-decreasing function $\phi\colon [0,1] \to [0,1]$. If ϕ satisfies

$$\int_0^1 \phi(x)\,dx = \frac{1}{2},\ \phi(0) = 0,\ \phi(1) = 1 \tag{13}$$

then η is (i) mean-preserving and (ii) priority-preserving.

The choice of the function ϕ now allows customizing the rate with which η adapts to different ranges of staleness.

Proof of Theorem A.1. (i) Mean-preservation follows from

$$\begin{aligned}\mathbf{E}[C_{s,\phi}(\tau)] &= \sum_{\tau=1}^{\infty} (1 + A \cdot (1 - 2\,\phi(F_{\tilde{\tau}}(\tau))))\,p_{\tilde{\tau}}(\tau) \\ &= 1 + A \cdot \left(1 - 2\sum_{\tau=1}^{\infty} \phi(F_{\tilde{\tau}}(\tau))p_{\tilde{\tau}}(\tau)\right)\end{aligned}$$

Let f extend p so that $f_{\tilde{\tau}}(t) = p_{\tilde{\tau}}(\tau)$ for $t \in (\tau-1, \tau)$. Then we have

$$\begin{aligned}\mathbf{E}[C_{s,\phi}(\tau)] &= 1 + A \cdot \left(1 - 2\int_{\tau=0}^{\infty} \phi(F_{\tilde{\tau}}(\tau))\,f_{\tilde{\tau}}(\tau)\,d\tau\right) \\ &= 1 + A \cdot \left(1 - 2\int_0^1 \phi(F_{\tilde{\tau}}(\tau))\,dF_{\tilde{\tau}}(\tau)\right) = 1\end{aligned}$$

(ii) Priority-preservation follows directly from that $F_{\tilde{\tau}}(\tau)$ is a CDF, hence non-decreasing, and the assumptions on A and ϕ. □

A.2. Proofs of Lemmas and Theorems that appear in the main part of the paper

In this section we present the proofs omitted in the main text.

Proof of Theorem 4.6. The statement is a special case, and follows from, Theorem A.1, with $\phi(\tau) = \tau$, which satisfies the requirements on $\phi(\tau)$ (13). □

Proof of Lemma 4.7. (i)-(ii) follow directly from Definition 4.5.

(iii):

$$\begin{aligned}Var[\eta(\tau)] &= \mathbf{E}\big[\eta(\tau)^2\big] - \bar{\eta}^2 \\ &= \sum_{\tau=1}^{\infty} (1 + A(1 - 2F(\tau))\bar{\eta})^2\,p_{\tilde{\tau}}(\tau) - \bar{\eta}^2 = \left(\frac{1}{6A}\,(1 + A(1 - 2F(\tau)))^3\Big|_{\tau=\infty}^{0} - 1\right)\bar{\eta}^2 \\ &= \left(\frac{1}{6A}\,\big((1+A)^3 - (1-A)^3\big) - 1\right)\bar{\eta}^2 = \frac{1}{3}(A\bar{\eta})^2\end{aligned}$$

(iv):

$$\begin{aligned}C_A(\tau) &= 1 + A(1 - 2F(\tau)) \\ &= 1 + A\,(2\,(1 - F(\tau)) - 1)\end{aligned}$$

Markov's inequality now gives

$$C_A(\tau) \leq 1 + A\left(2\frac{\mathbf{E}[\tau]}{\tau} - 1\right)$$

Now, assuming $\mathbf{E}[\tau] \approx m-1$, m being the number of threads, concludes the proof. Based on empirical studies, it has been observed that such an assumed condition is reasonable, generally holding in practice (Bäckström et al., 2019; Zhang et al., 2016). □

Proof of Lemma 5.3. From assumption 5.1 we have in particular

$$L(\theta_{t+1}) - L(\theta_t) - \langle \nabla L(\theta_t), \theta_{t+1} - \theta_t \rangle \leq \frac{\mathcal{L}}{2}\|\theta_{t+1} - \theta_t\|^2$$

From the SGD step we have

$$L(\theta_{t+1}) - L(\theta_t) - \eta_t\Big\langle \nabla L(\theta_t), -\nabla\tilde{L}(\theta_t) + \Big(\nabla\tilde{L}(\theta_t) - \nabla\tilde{L}(v_t)\Big)\Big\rangle \leq \frac{\mathcal{L}}{2}\eta_t^2\|\nabla\tilde{L}(v_t)\|^2$$

$$\Rightarrow L(\theta_{t+1}) - L(\theta_t) + \eta_t\left\langle \nabla L(\theta_t), \nabla\tilde{L}(\theta_t)\right\rangle - \eta_t\|\nabla L(\theta_t)\|\|\nabla\tilde{L}(\theta_t) - \nabla\tilde{L}(v_t)\|$$

$$\leq \frac{\mathcal{L}}{2}\eta_t^2\|\nabla\tilde{L}(v_t)\|^2$$

We have by assumption 5.1, and the triangle inequality

$$\|\nabla\tilde{L}(\theta_t) - \nabla\tilde{L}(v_t)\| \leq \mathcal{L}\|\theta_t - v_t\| = \mathcal{L}\left\|\sum_{i=1}^{\tau_t}\theta_{t-i+1} - \theta_{t-i}\right\| \leq \mathcal{L}\sum_{i=1}^{\tau_t}\eta_{t-i}\|\nabla\tilde{L}(\theta_{t-i})\|$$

$$\Rightarrow L(\theta_{t+1}) - L(\theta_t) + \eta_t\left\langle \nabla L(\theta_t), \nabla\tilde{L}(\theta_t)\right\rangle - \eta_t\mathcal{L}\|\nabla L(\theta_t)\|\sum_{i=1}^{\tau_t}\eta_{t-i}\|\nabla\tilde{L}(\theta_{t-i})\|$$

$$\leq \frac{\mathcal{L}}{2}\eta_t^2\|\nabla\tilde{L}(v_t)\|^2$$

Take expectation conditioned on the last staleness τ_t. From mean-independence, we have

$$\mathbf{E}[L(\theta_{t+1}) - L(\theta_t) \mid \tau_t] + \eta_t\mathbf{E}\big[\|\nabla L(\theta_t)\|^2\big] - \eta_t\mathcal{L}\sum_{i=1}^{\tau_t}\mathbf{E}[\eta_{t-i}]\mathbf{E}\Big[\|\nabla L(\theta_t)\|\|\nabla\tilde{L}(\theta_{t-i})\|\Big]$$

$$\leq \frac{\mathcal{L}}{2}\eta_t^2\mathbf{E}\Big[\|\nabla\tilde{L}(v_t)\|^2\Big] \leq \frac{\mathcal{L}}{2}M^2\eta_t^2$$

Applying Hölder's inequality, and that $\mathbf{E}[\tau_t] = \mathbf{E}[\tau]\ \forall t$

$$\mathbf{E}[L(\theta_{t+1}) - L(\theta_t) \mid \tau_t] + \eta_t\mathbf{E}\big[\|\nabla L(\theta_t)\|^2\big] - \eta_t\mathbf{E}[\eta]\mathcal{L}\sum_{i=1}^{\tau_t}\sqrt{\mathbf{E}[\|\nabla L(\theta_t)\|^2]\mathbf{E}\Big[\|\nabla\tilde{L}(\theta_{t-i})\|^2\Big]}$$

$$\leq \frac{\mathcal{L}}{2}\eta_t^2\mathbf{E}\Big[\|\nabla\tilde{L}(v_t)\|^2\Big] \leq \frac{\mathcal{L}}{2}M^2\eta_t^2$$

$$\Rightarrow \mathbf{E}[L(\theta_{t+1}) - L(\theta_t) \mid \tau_t] + \eta_t\mathbf{E}\big[\|\nabla L(\theta_t)\|^2\big] - \tau_t\eta_t\mathbf{E}[\eta]\mathcal{L}M^2 \leq \frac{\mathcal{L}}{2}M^2\eta_t^2$$

Now, take full expectation

$$\mathbf{E}[L(\theta_{t+1}) - L(\theta_t)] + \mathbf{E}[\eta]\mathbf{E}\big[\|\nabla L(\theta_t)\|^2\big] - \mathbf{E}[\tau\eta]\mathbf{E}[\eta]\mathcal{L}M^2 \leq \frac{\mathcal{L}}{2}M^2\mathbf{E}\big[\eta^2\big]$$

which concludes the proof. □

Proof of Theorem 5.6. From Corollary 5.5, we have

$$\mathbf{E}\big[\|\nabla L(\theta_t)\|^2\big] \leq \frac{\mathbf{E}[L(\theta_t) - L(\theta_{t+1})]}{\bar{\eta}} + \mathcal{L}M^2\bar{\eta}\left(\frac{A^2}{6} + \bar{\tau}\right)$$

$$\Rightarrow \frac{1}{T}\sum_{t=0}^{T-1}\mathbf{E}\big[\|\nabla L(\theta_t)\|^2\big] \leq \frac{\delta}{\bar{\eta}T} + \mathcal{L}M^2\bar{\eta}\left(\frac{A^2}{6} + \bar{\tau}\right)$$

which implies in particular that $\min_t \mathbf{E}\big[\|\nabla L(\theta_t)\|^2\big]$ satisfies the above upper bound as well. The bound now rewrites as in the statement by substituting $\bar{\eta}$ for (10), which concludes the proof. □

Proof of Theorem 5.8. With Corollary 5.5 as a starting point, we have by assumption 5.7

$$\mathbf{E}[L(\theta_{t+1}) - L(\theta_t)] + \bar{\eta}\mu\mathbf{E}[L(\theta_t) - L(\theta^*)] \leq \mathcal{L}M^2\bar{\eta}^2\left(\frac{A^2}{6} + \bar{\tau}\right)$$

$$\Rightarrow \mathbf{E}[L(\theta_T) - L(\theta^*)] \leq (1 - \bar{\eta}\mu)\mathbf{E}[L(\theta_{T-1}) - L(\theta^*)] + \mathcal{L}M^2\bar{\eta}^2\left(\frac{A^2}{6} + \bar{\tau}\right)$$

$$= (1 - \bar{\eta}\mu)^T \delta + \mathcal{L}M^2\bar{\eta}^2 \sum_{i=0}^{T-1}(1 - \bar{\eta}\mu)^i \left(\frac{A^2}{6} + \bar{\tau}\right)$$

$$\leq (1 - \bar{\eta}\mu)^T \delta + \frac{\mathcal{L}M^2\bar{\eta}}{\mu}\left(\frac{A^2}{6} + \bar{\tau}\right) < \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon$$

□

A.3. Convex convergence

For the sake of self-containment, we establish fundamental convex convergence bounds for (i) arbitrary staleness-adaptive step size functions, (ii) ones satisfying the ASAP.SGD properties, as well as (iii) the TAIL-τ function. For this, we will require also strong convexity:

Assumption A.2. L is strongly convex with parameter $\mathcal{C}$

$$\mathbf{E}\left[(x-y)^T\left(\nabla L(x) - \nabla L(y)\right)\right] \geq \mathcal{C}\|x-y\|^2 \; \forall x, y$$

Theorem A.3. *Consider the unconstrained optimization problem of (1). Under Assumptions 5.1, 5.2, and A.2, for any precision* $\epsilon > 0$*, and for any staleness-adaptive step size function* η *(Definition 4.1), there is a number* T *of* AsyncSGD *iterations, of the form (3) such that* $\mathbf{E}[\|\theta_T - \theta^*\|^2] < \epsilon$*, where* T *is bounded by:*

$$T \leq \frac{\ln\left(\|\theta_0 - \theta^*\|^2\epsilon^{-1}\right)}{2\left(\mathcal{C} - \mathcal{L}M\epsilon^{-1/2}\mathbf{E}[\tau\eta]\right)\bar{\eta} - \epsilon^{-1}M^2\mathbf{E}[\eta^2]}$$

Corollary A.4. *Under the same conditions as Theorem A.3, let* η_t *be* ASAP.SGD *adaptive step size function. Then we have the following bound on the expected number of iterations until expected convergence:*

$$T \leq \frac{\ln\left(\|\theta_0 - \theta^*\|^2\epsilon^{-1}\right)}{2\mathcal{C}\bar{\eta} - \epsilon^{-1}M\left(M + 2\mathcal{L}\sqrt{\epsilon}\bar{\tau}\right)\mathbf{E}[\eta^2]} \tag{14}$$

Corollary A.5. *Under the same conditions as Theorem A.3, let additionally* η_t *be* TAIL-τ *step size function. Then we have the following bound on the expected number of iterations until expected convergence:*

$$T \leq \frac{\ln\left(\|\theta_0 - \theta^*\|^2\epsilon^{-1}\right)}{2\mathcal{C}\bar{\eta} - \epsilon^{-1}M\left(M + 2\mathcal{L}\sqrt{\epsilon}\bar{\tau}\right)\left(1 + \frac{A^2}{3}\right)\bar{\eta}^2} \tag{15}$$

The proofs for Theorem A.3, and Corollary A.4, A.5 build on and extend the results in (Bäckström et al., 2019), and follow from the properties of ASAP.SGD and TAIL-τ, specified in Section 4.

A.4. Additional Empirical Results

Figure 7 provides an overview of the scalability of all algorithms evaluated here for MNIST and Fashion-MNIST, respectively, over a large parallelism spectrum. We observe speedup until around 32 threads, at which the system saturates and does not benefit from higher parallelism. The different algorithms perform differently at different parallelism levels. In particular, under hyper-threading ($m > 36$), the *AsyncSGD* implementations suffer a slower convergence due to increased computational overhead and asynchrony-induced noise. However, in all instances, TAIL-τ provides significant speedup, independently of the algorithm, dataset, and parallelism level.

Figure 8 and 9 show staleness distributions of the considered *AsyncSGD* algorithms for LeNet and MLP training, respectively, together with the corresponding scaling factors C_A of the staleness-adaptive TAIL-τ step size functions. The *AsyncSGD* implementations have fundamentally different staleness distributions, due to the underlying algorithmic mechanisms for progress and consistency. Moreover, we observe multi-modality in some executions, in particular under hyper-threaded parallelism ($m > 36$), and for lock-based *AsyncSGD* due congestion about the locks. The execution-specific staleness distributions are utilized by TAIL-τ to generate an instance-based adaptiveness strategy, accommodating for algorithmic differences and underlying system aspects, enabling the improvements in convergence rates and stability that are observed in Section 6.

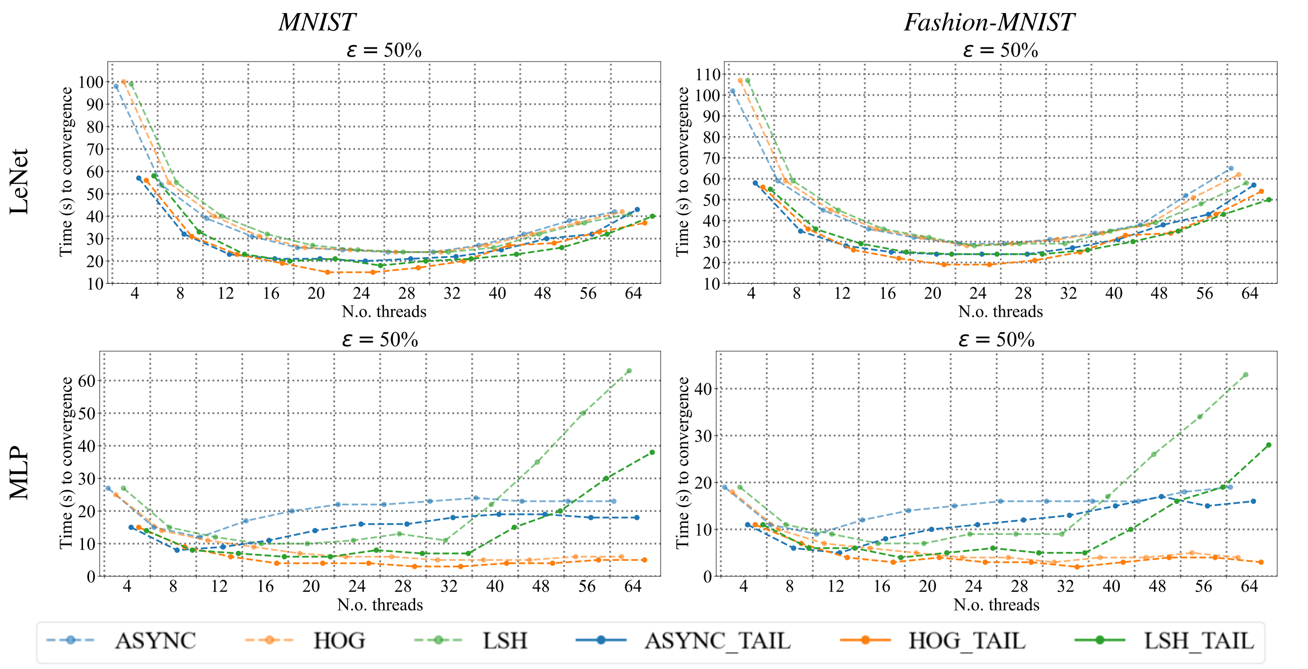

Figure 7. *AsyncSGD* convergence rates over a wide parallism spectrum, inluding hyper-threading ($m > 36$), for HOGWILD! (HOG), *Leashed-SGD* (LSH), and lock-based (ASYNC), comparing executions using the staleness-adaptive TAIL-τ (suffix: _TAIL) against standard, constant step size.

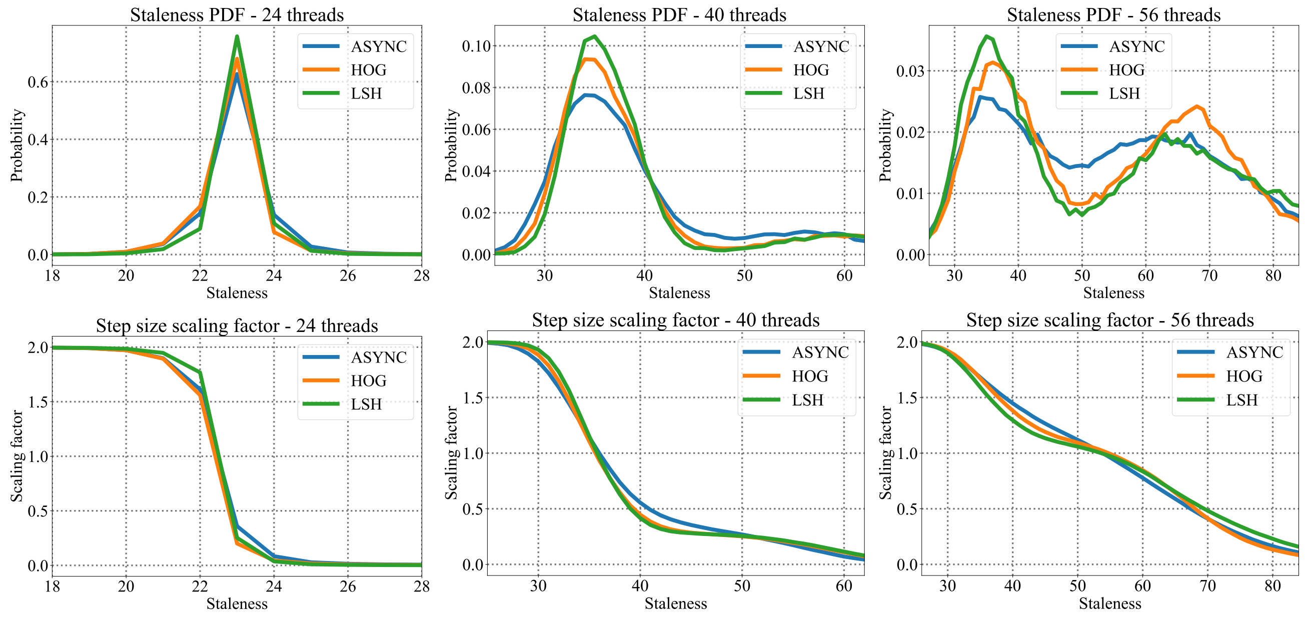

Figure 8. Staleness distributions for the considered *AsyncSGD* algorithms (*upper row*) and the corresponding scaling factors C_A of the generated staleness-adaptive TAIL-τ step sizes, as in Definition 4.1 (*bottom row*) for LeNet training on MNIST and Fashion-MNIST.

A.5. Evaluation of staleness-based static dampening

Figure 10 and 6 show convergence rates and training plots under varying parallelism for the $\times\tau^{-1}$ scheme which, with some variation, appears often in previous works (Sra et al., 2016; Zhang et al., 2016; Ren et al., 2020), compared to standard *AsyncSGD* with constant step size.

For both LeNet and MLP training for MNIST recognition, we observe significant challenges with achieving convergence rates within the same order of magnitude as traditional, constant step size, *AsyncSGD*. This, in contrast with TAIL-τ which,

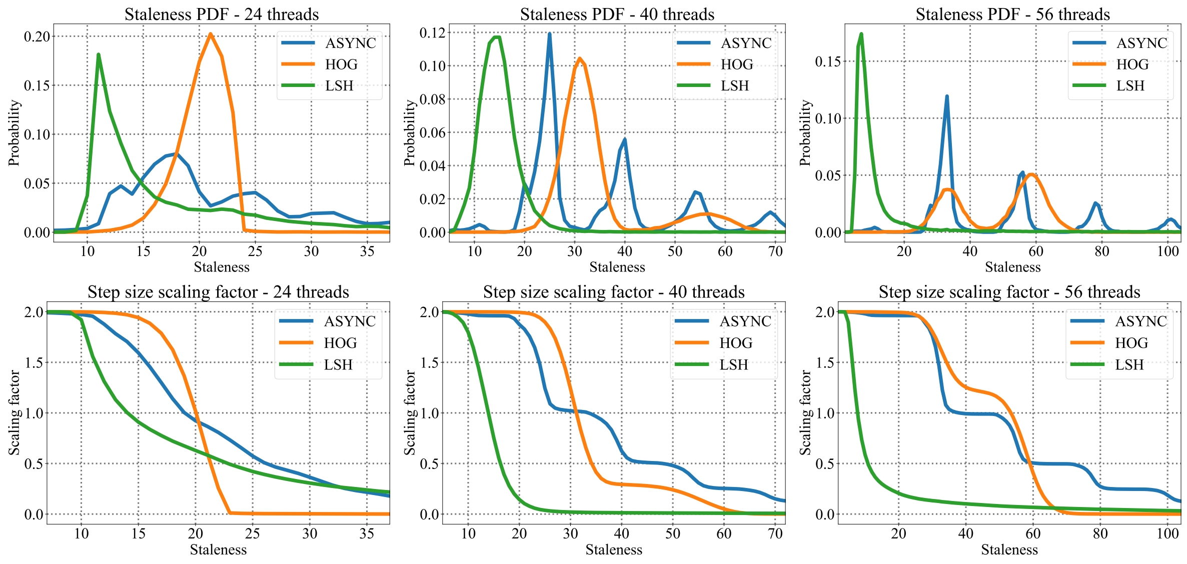

Figure 9. Staleness distributions for the considered *AsyncSGD* algorithms (*upper row*) and the corresponding scaling factors C_A of the generated staleness-adaptive TAIL-τ step sizes, as in Definition 4.1 (*bottom row*) for MLP training on MNIST and Fashion-MNIST.

as shown in Section 6, provides persistent, and significant, speedup in convergence rates. The straight-forward $\times\tau^{-1}$ step size suffers significant challenges in achieving convergence, especially under higher parallelism, as explained in Section 3.

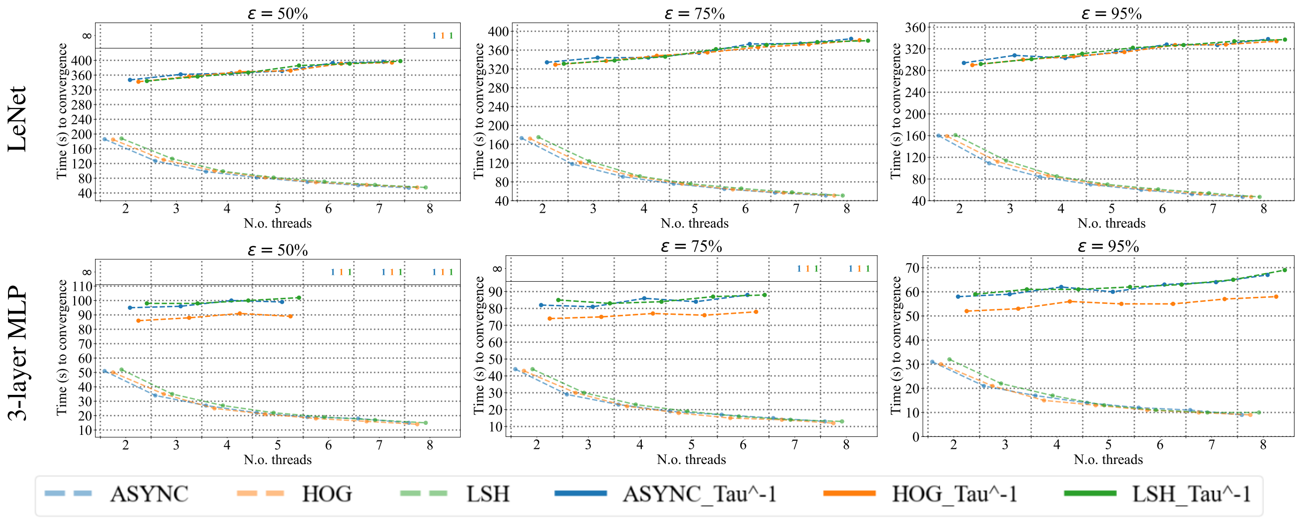

Figure 10. Convergence rates on MNIST for LeNet and a 3-layer MLP with *AsyncSGD*, with HOGWILD! (HOG), *Leashed-SGD* (LSH), and traditional lock-based (ASYNC) implementations, comparing executions using the $\times\tau^{-1}$ staleness-adaptive scheme (suffix: _Tau∧-1) against standard, constant step size.