We would like to thank all reviewers for their comments and questions.

**Reviewer 2:** We appreciate your recommendation about reordering the paper. We chose the current ordering because we thought that a detailed discussion of the framework early on may make the paper somewhat less accessible to researchers not directly familiar with the main challenges in the design and analysis of asynchronous algorithms. Yet, we agree that the current ordering does not fully distill the view that a more experienced reader may appreciate. Thus, in the revised version, we will present the crux of the generic framework (e.g., Section 5, which should still be accessible to the general reader) early on, before Sections 3 & 4, and add further remarks about the design choices as appropriate.

**Reviewer 3:** Thanks for your positive comment. It is indeed correct that the regret bounds imply delayed-feedback results for online learning. In particular, unlike the asynchronous setting, in delayed-feedback online learning (and parameter-server models) one can typically assume that $\tau_t$ is known at time $t$. Then, using Theorem 5 with $\nu_t = \tau_t$, the penalty term $p_*\nu_t + \sum_{s \in O_t} \frac{\tau_s}{\nu_s}$ is bounded by $\tau_t + \tau_t'$, even when $p_* = 1$. The resulting regret bound extends the delay-dependent adaptive results in the literature (e.g., Ref [14] given in the paper, or [Joulani et al, "Delay-tolerant online convex optimization: Unified analysis and adaptive-gradient algorithms", AAAI-2016]), and this close connection makes it possible to extend their delay-adaptive tuning techniques to a larger set of problems.

**Reviewer 4:** Thanks for looking into the details of our paper. We have answered your questions below:

**Linear speed-up.** We agree that the definition of obtaining a linear speed-up here is somewhat specific to the community; we are here using the methodology established in previous work (e.g., [Recht et al, "Hogwild: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent", 2011]). We have recalled the basics of the definition on Lines 31-38 of the paper, but we will expand the discussion in the final version of the paper to make it more clear.

To illustrate the definition, we first re-call the generic parallel computing definition: given a serial algorithm that solves a given problem, a parallelized version of the algorithm is said to achieve "linear speed-up" if it solves the same problem $(c \times P)$-times faster (in wall-clock time) than the serial algorithm, where $c \leq 1$ is a constant and $P$ is the number of processors used. For example, in a so-called "embarrasingly parallel" problem (like hyper-parameter search) where the problem can be split into smaller, independent sub-problems, a parallel algorithm can achieve a linear speed up with $c \approx 1$ by simply assigning different sub-problems to the $P$ processors and solving them simultaneously.

In case of a parallel asynchronous optimization algorithm, the key point is that linear speed-ups may not be possible in general, except in specific "regimes" when the problem is, e.g., sparse enough, and the number of processors is not too large. Theorems 1 and 2 in particular quantify this for ASYNCADA and HEDGEHOG: if for some $c'$, we have $\tau_*^2 \leq c'/p_*$, then to achieve the same accuracy as a serial algorithm would achieve after $T$ updates, ASYNCADA needs to make $(1 + c')T$ updates (since accuracy $\approx R_T/T$, c.f. lines 104-111 of the paper). Since ASYNCADA makes updates $P$ times faster (in wall-clock time, by doing $P$ updates in parallel), this implies that ASYNCADA solves the same problem (achieves the same accuracy) $P/(1 + c')$ times faster than the serial algorithm would, hence enjoying a "linear speed-up" with $c = 1/(1 + c')$ under the sparse-data regime. Importantly, the speed-up regime $p_*\tau_*^2 = O(1)$ is the same as what was established by the previous work of Refs [10, 11, 18] of the paper.

**Definition of $p_*$.** Your example is correct. In such a case, the problem is simply not sparse, since the gradient coordinates are not zero with positive probability, and our result does not lead to a linear speed-up (neither do the results from previous work, with the exception of [10,11] for box-shaped, unbounded, non-proximal SGD/SAGA, with general linear speed-ups without sparsity remaining an open problem [18]). However, our regret bounds still hold, in terms of the observed delays $\tau_t, \tau_t'$, even when $p_* = 1$, and generalize the delay-adaptive regret bounds of online optimization (as referenced in response to Reviewer 3) to inconsistent delays and coordinate perturbations.

**Remark 3.** We will rephrase this remark to make it more clear. The aim of the remark is to compare part (iii) of Theorem 1 to the work of Nguyen et al. [25]. Nguyen et al. [25] recognize the fact that for a strongly-convex objective on an unbounded domain, one cannot assume that the gradients are uniformly bounded by a constant $G_*$ (the "bounded-gradient" assumption). Hence, they provide an analysis of strongly-convex optimization using unconstrained Hogwild! with a global clock, without relying on the bounded-gradient assumption. The aim of Remark 3 is to emphasize that the $G_*$ bound in part (iii) of Theorem 1 is not a global upper-bound on the gradients of a strongly-convex function: it is a global upper-bound only on the non-strongly-convex part of the function, and thus compatible with strong convexity. Hence, like Nguyen et al. [25], Theorem 1 (iii) avoids the incompatible bounded-gradients assumption when the objective is strongly-convex, but further applies to any constraint set without requiring a global clock.

**Typos and errors.** We will fix all the mistakes on Pages 1 and 2 and perform a further proofreading of the paper, as well as the definition of $\hat{t}$ on Line 5 of Alg 1. Thanks for the catch!