WHEN IS OPTIMIZER MEMORY FEDERABLE? ADAPTIVE SYNCHRONIZATION IN FEDERATED LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Federated learning methods increasingly share optimizer states across clients, yet these states are synchronized on the model's clock or on a schedule fixed by the method. We ask which optimizer memories should be federated, and how often. Decoupling optimizer-memory synchronization from model synchronization, we distinguish *federability*, the immediate benefit of replacing a client's memory with the aggregate, from the long-run utility of a synchronization policy. Fixed-interval experiments show that Adam's second-moment tolerates long intervals, whereas the FedMuon policy we evaluate degrades once the interval is expanded under strong heterogeneity, largely through the coupling of momentum delivery with its global-direction correction. For Adam, an exact first-step analysis and measurements on training runs show that the aggregate effect of transporting the second-moment depends on the joint variation of client memory deviations and optimizer response rather than on memory discrepancy alone, and that the realized gain follows the signed alignment between the endpoint difference and the evaluation-loss gradient. We therefore introduce a matched transport-gain probe that compares short-horizon training with local versus aggregated memory, and adapt each memory's interval from its outcome. On CIFAR-10 and CIFAR-100, the controller recovers the fixed-interval sensitivities and reduces uplink by up to 48%, including probe traffic, while staying within about one point of every-round synchronization. Our code is available at anonymous repository.

1 INTRODUCTION

Federated learning (FL) alternates local optimization with communication to train a shared model without centralizing client data (McMahan et al., 2017). Communication schedules in FL are typically defined around model synchronization, but model parameters are not the only quantities that persist across local optimization steps. Modern optimizers maintain internal states that accumulate optimization history. Adam, for example, maintains first- and second-moment estimates, while Muon maintains momentum that is transformed via orthogonalization (Kingma & Ba, 2014; Jordan et al., 2024). Recent methods show that these optimizer states can themselves be federated. For example, FedAdamW (Liu et al., 2026) communicates block-mean second-moment information, while FedMuon (Liu et al., 2025) federates momentum alongside a local–global direction correction.[1]

Federating optimizer state raises a separate question: *how often should they be synchronized?* There is no a priori reason for this schedule to coincide with the model communication clock. Different optimizer states encode distinct statistics and evolve under different recurrences, so the useful information they share across clients may persist over different timescales. We refer to these persistent optimizer states as *optimizer memories*, emphasizing that they carry information accumulated over the optimization trajectory.

Related work in distributed data-parallel (DDP) training has similarly shown that optimizer states need not synchronize on a single schedule (Iacob et al., 2026). Translating this idea to FL, however, introduces a different challenge. Heterogeneous client objectives and variable participation cause optimizer memories to become client-specific between synchronization rounds. Choosing when to

[1] Momentum is aggregated before orthogonalization, since an average of orthogonal matrices is generally not orthogonal.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

synchronize a memory must therefore balance two competing risks, where over-synchronization incurs unnecessary communication, whereas under-synchronization allows useful shared state to diverge or become stale. The appropriate balance can depend on data heterogeneity (Karimireddy et al., 2020; Li et al., 2020) and participation dynamics (Jhunjhunwala et al., 2022; Wu et al., 2023). Yet it remains unclear how the appropriate synchronization timescale varies across optimizer memories and federated environments.

We therefore ask: **which optimizer memories should be federated, and at what timescale?** We separate two aspects of this question. We define *federability* as the immediate benefit of replacing a client's local optimizer state with an aggregated state, and distinguish it from *policy utility*, the long-run effect of repeatedly applying synchronization. We study these questions using Muon momentum M and Adam's *full-coordinate* second-moment v, where full-coordinate denotes the complete element-wise state and distinguishes it from FedAdamW's block-mean representation. Throughout our study, model parameters are synchronized with FedAvg (McMahan et al., 2017) every round, while varying optimizer-memory synchronization independently. This isolates the synchronization timescale of optimizer memory from the model communication clock.

Our experiments show that the two memories exhibit distinct useful synchronization timescales. Adam's second-moment v is comparatively tolerant to longer synchronization intervals. The FedMuon policy we evaluate is far more sensitive to interval expansion under high data heterogeneity, and a controlled ablation attributes much of that sensitivity to the coupling between momentum delivery and the global-direction correction rather than to momentum delivery alone. Thus, a synchronization interval that is adequate for one memory or heterogeneity regime can be unnecessarily frequent or insufficient for another.

Motivated by this variation, we ask whether the need for synchronization can be diagnosed online rather than specified in advance. We introduce a short-horizon *transport-gain* probe that estimates immediate federability by measuring the realized benefit of transporting an aggregated optimizer memory to a client relative to retaining its local memory. Our analysis characterizes Adam's first-step response to memory transport and the loss effect of the realized branch endpoints (Section 4.3), motivating the evaluation-loss contrast the probe measures. We use this signal to adapt the synchronization interval during training, retaining frequent synchronization when fresh shared memory is useful and increasing the interval when it is not. The resulting online decisions align with the memory- and heterogeneity-dependent sensitivities observed under fixed synchronization intervals.

Our contributions are fourfold:

1. We formulate optimizer-memory federation as a component-specific synchronization problem, separating the optimizer-memory clock from the model clock and distinguishing the immediate *federability* of a memory from the long-run *utility* of a policy.
2. We show that synchronization sensitivity varies across optimizer-memory mechanisms and federated regimes, using Adam's full-coordinate second moment v and FedMuon momentum M as two cases.
3. We characterize the mechanism behind Adam transport gain, connecting client-specific responses to the aggregate update effect and the realized endpoint difference to its evaluation-loss effect.
4. We develop a matched transport-gain probe for the realized benefit of synchronization and use it to adapt intervals online; the resulting decisions match the sensitivities observed under fixed schedules.

2 Problem Formulation and Hypothesis

We study whether an optimizer memory x should be shared across clients, and how often, while the model synchronizes every round. We denote its synchronization interval by K_x, where $K_x = 1$ aggregates the memory every round and $K_x = \infty$ keeps it strictly local.

Because optimizer memories accumulate local gradient history, client memories drift apart during local steps. Data heterogeneity accelerates this drift, and partial participation changes which clients form the aggregate. The benefit of transporting an aggregated memory, and how long that benefit lasts, should therefore depend on the federated setting.

Hypothesis: The effective synchronization interval of an optimizer memory depends on the realized federated setting and on how the memory enters the optimizer update. Drift between client memories alone does not determine the utility of transporting them.

We test this hypothesis in two complementary ways, where fixed-K_x experiments measure *policy utility*, the long-run effect of an interval, and transport-gain probes measure *federability*, the immediate benefit of transport at a given round. The two can disagree, as a memory whose trajectory degrades under large K_x may still show a large short-term transport gain.

We apply both measurements to Adam's second moment v and to Muon's momentum M. For v, we federate the full-coordinate state rather than the block-mean summary communicated by FedAdamW; Appendix G compares the two. For Muon, we follow the FedMuon of Liu et al. (2025), which aggregates client momentum and thereby makes it a federated memory, rather than the method of the same name by Takezawa et al. (2026), which corrects the bias of Muon's linear minimization oracle. Adam's first moment m is kept local and reset at the start of each local segment throughout.

3 Why the Federated Setting Changes Memory Utility

We narrow the hypothesis to how local optimization changes the state that is later aggregated and transported.

Data heterogeneity induces optimizer-memory heterogeneity. Heterogeneous client data induce different local objectives and, consequently, different gradient trajectories even when clients start from the same model. For Adam's second moment, let $v_{i,h}^r$ and $g_{i,h}^r$ denote client i's state and gradient at local step h of round r, let β_2 denote the decay, and let $(\cdot)^{\odot 2}$ denote elementwise squaring. Two clients i and j that perform the same number H of local steps satisfy

$$v_{i,H}^r - v_{j,H}^r = \beta_2^H (v_{i,0}^r - v_{j,0}^r) + (1-\beta_2) \sum_{h=1}^{H} \beta_2^{H-h} \left((g_{i,h}^r)^{\odot 2} - (g_{j,h}^r)^{\odot 2} \right). \tag{1}$$

Different gradient trajectories thus produce divergent second-moment memories. Similarly, Muon momentum accumulates local gradient history before mapping it to the update direction. See Appendix A for complete update rules for Adam, FedAdamW, Muon, and FedMuon.

Participation changes the transported aggregate. Let $\mu_x^r = N^{-1} \sum_{j=1}^N x_j^r$ denote the mean of the client memory states x_j^r over all N clients, and let $\hat{\mu}_x^r$ denote their mean over the qN clients that participate, sampled uniformly without replacement, where q is the participation rate. If

$$\Sigma_x^r = \frac{1}{N} \sum_{j=1}^{N} (x_j^r - \mu_x^r)(x_j^r - \mu_x^r)^\top, \quad \text{then} \quad \mathbb{E}\left[\|\hat{\mu}_x^r - \mu_x^r\|^2 \right] = \frac{1-q}{q(N-1)} \operatorname{tr}(\Sigma_x^r). \tag{2}$$

Lower participation therefore increases aggregate sampling error when client states are dispersed. Thus, heterogeneity changes the memories being aggregated (Eq. (1)), and partial participation changes the aggregate being transported (Eq. (2)).

Contrast with synchronous DDP. Synchronous DDP aggregates gradients at every step with all workers participating, so neither effect arises there.

4 Methodology

4.1 Optimizer-Memory Synchronization

We consider N clients with local objectives $F_j(w)$ and a shared model w. At round r, participating clients $\mathcal{S}_r$ optimize locally from w^r, and the model is aggregated with FedAvg, so the model synchronization interval is one round throughout.

We study Adam's second-moment v and Muon's momentum M. Adam updates $v_{j,h}^r = \beta_2 v_{j,h-1}^r + (1-\beta_2)(g_{j,h}^r)^{\odot 2}$, while Muon maintains a momentum buffer that is transformed into an orthogonalized update direction. FedMuon additionally blends the local direction with a global direction derived

from the aggregated model updates (Liu et al., 2025). In the configuration we evaluate, this correction is applied on the rounds when aggregated momentum is delivered (Section 5).

Each optimizer memory $x \in \{v, M\}$ has its own synchronization interval K_x. When x is synchronized, the participating client states are aggregated as

$$\bar{x}^r = \frac{1}{|\mathcal{S}_r|} \sum_{j \in \mathcal{S}_r} x_j^r \tag{3}$$

and transported back to the clients. This separates the optimizer-memory clock from the every-round model clock.

4.2 Measuring Optimizer-Memory Transport Gain

At probe rounds, we measure the immediate effect of transporting the aggregated optimizer memory. Each client follows two matched branches of H local optimizer steps from the same initial model state. The *local* branch keeps its current optimizer memory, while the *transport* branch replaces only the probed memory with $\bar{x}^r$. The branches also share the minibatch sequence, hyperparameters, and all non-probed optimizer states, and differ only in the probed memory at initialization. Gradients and optimizer states then evolve separately along the two trajectories.

Let $\Delta w_{j,\mathrm{local}}^{r,H}$ and $\Delta w_{j,\mathrm{transport}}^{r,H}$ denote the resulting model updates. For branch $B \in \{\mathrm{local}, \mathrm{transport}\}$, we define

$$\overline{\Delta w}_B^{r,H} = \frac{1}{|\mathcal{S}_r|} \sum_{j \in \mathcal{S}_r} \Delta w_{j,B}^{r,H}. \tag{4}$$

Let L_{eval} denote the fixed evaluation function. The aggregate transport gain is

$$G_r = L_{\mathrm{eval}}\left(w^r + \overline{\Delta w}_{\mathrm{local}}^{r,H}\right) - L_{\mathrm{eval}}\left(w^r + \overline{\Delta w}_{\mathrm{transport}}^{r,H}\right). \tag{5}$$

A positive G_r means that transporting the optimizer memory produces a lower short-horizon aggregate loss. We define the transport-benefit indicator as $b_r = \mathbb{1}[G_r > 0]$. This measures *snapshot federability*, not the long-run quality of a fixed synchronization policy.

Each probe costs two additional H-step runs per participating client, the upload of both branch updates, and two evaluations of L_{eval} on a held-out server validation set (Appendix C quantifies this cost). Consequently, probing frequency is controlled separately from synchronization frequency (Section 4.4).

4.3 What Determines Transport Gain?

The transport gain G_r measures the realized benefit of replacing local optimizer memory with its aggregate. We analyze how this gain arises in two stages. Memory transport first changes the client updates and, after aggregation and local optimization, produces a parameter-space difference D_r^H. The effect of this difference on G_r then depends on the geometry of the evaluation loss: D_r^H records what transport changed, and G_r whether that change helped. We characterize both stages for Adam's second moment, beginning with how memory replacement changes the aggregate update.

Proposition 1 (Immediate response to transporting v)**.** *Under the first-step conditions of Appendix B, let $d_i = \Delta w_{i,\mathrm{transport}}^{r,1} - \Delta w_{i,\mathrm{local}}^{r,1}$ denote the transport-minus-local difference of the first-step model updates. Then*

$$d_i = s_i \odot (\bar{v}^r - v_i^r), \qquad D_r^1 = \frac{1}{|\mathcal{S}_r|} \sum_{i \in \mathcal{S}_r} d_i = \frac{1}{|\mathcal{S}_r|} \sum_{i \in \mathcal{S}_r} (s_i - \bar{s}) \odot (\bar{v}^r - v_i^r), \tag{6}$$

exactly in every coordinate, where s_i is the Adam response coefficient defined in Eq. (A5) *and $\bar{s}$ is its mean over the same clients.*

The proposition separates two factors in the immediate transport response, namely the client's memory deviation $\bar{v}^r - v_i^r$ from the participating-client aggregate, and the sensitivity s_i of its Adam update to that replacement, which depends on the client's current gradient and optimizer state. Because the

deviations sum to zero over the participating clients, the client-common component of s_i drops out after aggregation, and in each coordinate D_r^1 is the negative empirical covariance between s_i and v_i^r. The immediate aggregate response is therefore determined jointly by the memory deviations and the client-specific responses rather than by memory dispersion alone. Appendix B gives a two-client example in which identical incoming memories yield different aggregate responses.

From update response to measured gain. Proposition 1 is exact for the first matched step, whereas the probe follows both branches through the full local segment of H steps, along which local optimization reshapes the first-step difference (Section 6.2). We therefore work with the realized endpoint difference. For the full H-step probe, define

$$w_{\text{local}}^{r,H} = w^r + \overline{\Delta w}_{\text{local}}^{r,H}, \qquad D_r^H = \overline{\Delta w}_{\text{transport}}^{r,H} - \overline{\Delta w}_{\text{local}}^{r,H}. \tag{7}$$

The transport endpoint is $w_{\text{local}}^{r,H} + D_r^H$, with D_r^H taken from the two realized trajectories. We next characterize how this realized parameter-space response affects the quantity the controller observes.

Proposition 2 (Loss effect of the realized transport difference)**.** *Let L_{eval} be differentiable on a neighborhood of the segment from $w_{\text{local}}^{r,H}$ to $w_{\text{local}}^{r,H} + D_r^H$, with $\|\nabla L_{\text{eval}}(w_{\text{local}}^{r,H} + uD_r^H) - \nabla L_{\text{eval}}(w_{\text{local}}^{r,H})\|_2 \le Lu\|D_r^H\|_2$ for $0 \le u \le 1$ and some finite $L \ge 0$. Then*

$$G_r = -\big\langle \nabla L_{\text{eval}}(w_{\text{local}}^{r,H}), D_r^H \big\rangle + R_r, \qquad |R_r| \le \frac{L}{2}\|D_r^H\|_2^2. \tag{8}$$

The first-order term is the alignment of D_r^H with the negative evaluation-loss gradient at the local endpoint and certifies the sign of G_r whenever it exceeds the remainder bound. Together, the two propositions separate three levels of optimizer-memory transport: memory-state variation, the induced parameter-space response, and its loss-space effect. The controller targets the last of these. Rather than estimating these intermediate quantities separately, the matched probe measures their realized consequence directly through G_r. Section 6.2 evaluates both propositions on the training runs.

4.4 Adaptive Synchronization Using Transport Gain

Section 4.3 motivates the measured loss contrast G_r as the synchronization signal. Each memory x has a delivery interval K_x, which sets how often it is synchronized, and a probing interval J_x, which sets how often G_r is measured. We initialize $K_x = 1$ (every-round synchronization) and $J_x = 1$. Memory is synchronized every K_x rounds, and each probing round, held J_x rounds after the previous one, computes G_r (Eq. (5)) and $b_r = \mathbb{1}[G_r > 0]$. The intervals are then updated as

$$K \leftarrow \phi(K, b) = \begin{cases} 2K, & b = 0, \\ K, & b = 1, \end{cases} \qquad J \leftarrow \begin{cases} K, & \text{if } K \text{ changed}, \\ 2J, & \text{otherwise}, \end{cases} \tag{9}$$

with $K = K_x$, $J = J_x$, and $b = b_r$. Each negative gain doubles K_x, and each positive gain keeps it, so synchronization becomes rarer but never stops. Since the right interval may lie anywhere between one round and the length of training, this exponential backoff searches on a logarithmic scale, reaching K after about $\log_2 K$ negative gains. Starting from every-round synchronization places the controller's errors on the side of extra communication rather than lost accuracy, which matters because under-synchronization can cost far more (Section 6.1). It also avoids deciding whether to start synchronizing from the noisy gains of the earliest rounds.

The measured gains determine K_x, but probing need not follow the synchronization schedule. If probing accompanied every synchronization when transport is useful ($K_x = 1$), the controller would probe every round for the rest of training, repeatedly confirming the same decision at full probing cost. The only way to probe less would be to lengthen K_x, making synchronization rarer precisely when it helps. The probing interval J_x decouples these two schedules. When a gain leaves K_x unchanged, J_x doubles; in this scenario, synchronization continues every round while probing occurs only at rounds $1, 3, 7, 15, \ldots$. When a gain changes K_x, J_x resets to K_x, so the new interval is verified before probing backs off. Consequently, $J_x \ge K_x$ throughout, and every probing interval contains at least one synchronization round. Because K_x changes at most $\sim \log_2 R$ times and J_x doubles

between changes, total probing rounds over R rounds grow only polylogarithmically. Algorithm 1 details this complete procedure, which we term the *transport-gain controller*.

This controller is a simple heuristic rather than an optimal policy. It uses only the sign of G_r, never shortens an interval, never disables synchronization entirely, and ignores the communication cost of the memory. Designing better interval policies, for example by scaling the update with the magnitude of G_r, allowing contraction, or trading gain against communication cost, is a direction for future work.

Algorithm 1 Adaptive optimizer-memory synchronization from transport gain

Require: Total rounds R
$K \leftarrow 1$ {synchronization interval}
$J \leftarrow 1$ {probing interval}
$r_{\text{probe}} \leftarrow 1$ {next probing round}
for $r = 1, \ldots, R$ **do**
  Local training on $\mathcal{S}_r$; aggregate weights with FEDAVG
  if $r = r_{\text{probe}}$ **then** {probing round}
    Compute G_r {loss reduction from transporting memory, Eq. (5)}
    if $G_r \leq 0$ **then**
      $K \leftarrow 2K$; $J \leftarrow K$
    else
      $J \leftarrow 2J$
    end if
    $r_{\text{probe}} \leftarrow r + J$
  end if
  if $r \bmod K = 0$ **then**
    Aggregate and synchronize optimizer memory
  end if
end for

5 EXPERIMENTAL SETUP

Benchmarks. We train ResNet-18 (He et al., 2016) on CIFAR-10 (Krizhevsky & Hinton, 2009) with 10 clients and full participation for 100 rounds, and on CIFAR-100 with 100 clients and $q = 10\%$ participation for 300 rounds, using 50 local steps per round. We partition data with a Dirichlet distribution with $\alpha \in \{0.1, 0.6\}$, where smaller α means stronger heterogeneity. We repeat the adaptive experiments with ViT-Tiny (Kolesnikov et al., 2021; Steiner et al., 2021) on CIFAR-100. We give further details in Appendix C.

Transport diagnostics. On the CIFAR-10 Adam runs, we additionally record, without altering training, the quantities of Section 4.3 at six rounds per run, running both probe branches for the full $H = 50$ steps and evaluating their endpoint losses on a fixed set of 2048 training examples under evaluation transforms, using the same aggregate endpoint-loss contrast as the controller. This gives 120 snapshots over 20 runs, covering $\alpha \in \{0.1, 0.6\}$ and $K_v \in \{4, \infty\}$ with five seeds each.

Synchronization. We aggregate model parameters with FedAvg every round and vary only the optimizer-memory interval K_x. For Adam, we reset m at the start of each local segment and never federate it, and we aggregate and redistribute v at synchronization rounds. For Muon, synchronization delivers the aggregated momentum, and the model-derived global-direction correction (Liu et al., 2025) is applied on those delivery rounds, so varying K_M changes the frequency of momentum delivery and of the correction together. Between deliveries, clients train with local momentum.

Fixed and adaptive intervals. On the ResNet-18 benchmarks, we sweep $K_x \in \{1, 2, 4, 16, \infty\}$, where $K_x = \infty$ keeps the memory local. For ViT-Tiny, we evaluate only $K_x = 1$ and $K_x = \infty$. Adaptive runs follow Algorithm 1 from $K = 1$ and $J = 1$, with $H = 50$ probe steps and L_{eval} computed on a fixed server-held validation split, with a class-balanced 10% of the training data removed before client partitioning. We report the final interval $\bar{K}_x$ averaged over seeds, final test accuracy, and uplink per round. We compute uplink analytically, sum it over participating clients, and include probe traffic.

6 EVALUATION

6.1 OPTIMIZER MEMORIES EXHIBIT DIFFERENT SYNCHRONIZATION SENSITIVITIES

If the information a memory shares across clients persists over rounds, accuracy should stay flat as K_x grows; if it goes stale quickly, accuracy should drop as soon as $K_x > 1$. Heterogeneity should sharpen this contrast, since client memories drift apart faster (Eq. (1)). For example, at $\alpha = 0.1$, a CIFAR-10 client's data is typically dominated by a few classes, so with $K_x = 4$ it spends three of every four rounds on a memory shaped by those classes alone. Under mild heterogeneity ($\alpha = 0.6$), neither memory is sensitive: across all intervals, including $K_x = \infty$, accuracy varies by at most 0.5 points on CIFAR-10 and about 1 point on CIFAR-100 (Figure 1).

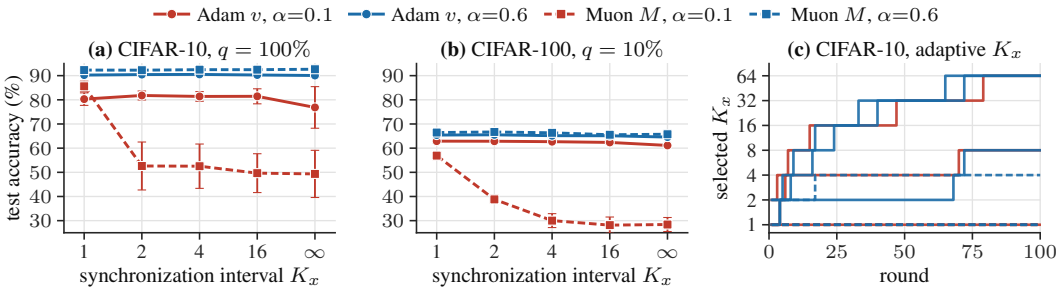

Figure 1: **(a), (b)** Final test accuracy (%) under fixed optimizer-memory synchronization intervals K_x for Adam v and Muon M on CIFAR-10 ($q = 100\%$) and CIFAR-100 ($q = 10\%$) with ResNet-18, under strong ($\alpha = 0.1$) and mild ($\alpha = 0.6$) heterogeneity, where q denotes the client participation rate. **(c)** Interval K_x selected by the transport-gain controller over training on CIFAR-10, one line per seed. Overlapping lines coincide.

Under strong heterogeneity ($\alpha = 0.1$), the two memories follow the two patterns. Adam v tolerates long intervals, with every finite K_v yielding 80.29–81.80% on CIFAR-10 and 62.37–62.90% on CIFAR-100, and accuracy drops only without synchronization (76.85% and 61.13%). The evaluated FedMuon policy degrades as soon as the interval grows, which Section 6.2 traces to its correction schedule. Accuracy falls from 85.60% at $K_M = 1$ to 52.62% at $K_M = 2$ on CIFAR-10, and more gradually on CIFAR-100, from 56.91% to about 28% for $K_M \geq 16$.

Takeaway: Sensitivity depends on both the memory and the regime. Adam v needs only occasional synchronization, whereas the evaluated FedMuon policy needs every-round synchronization under strong heterogeneity but not under mild heterogeneity. No single fixed interval suits all cases, which motivates choosing it online.

6.2 TRANSPORT GAIN ON REALIZED TRAINING STATES

The transport diagnostic of Section 5 shows what determines the effect of transporting Adam's second moment. Figure 2 summarizes its 120 snapshots, and Appendix D reports the condition-level values. Figure 2(a) asks whether large client-level responses survive aggregation. It plots the mean client response $\frac{1}{|\mathcal{S}_r|}\sum_i \|d_i\|_2$ against the aggregate response $\|D_r^1\|_2$ that the server applies. In every condition and across two decades of magnitude, the points lie well below the identity, with the aggregate typically 36–40% of the mean client response (median ratio 0.37 over snapshots), so client responses to transport partly cancel after aggregation, as the covariance form of Eq. (6) allows. Memory dispersion alone does not characterize the aggregate response (Appendix D).

Figure 2(b) quantifies the change from the first step to the realized endpoint: the median $\cos(D_r^1, D_r^h)$ falls to 0.22 at $h = 50$ while the magnitude grows by a median factor of 62, so D_r^H must be taken from the realized trajectories. Proposition 2 separates the signed term $A_r = -\langle \nabla L_{\text{eval}}(w_{\text{local}}^{r,H}), D_r^H \rangle$ from a remainder R_r. Figure 2(c) plots A_r against the measured gain over all 120 snapshots. The two lie along the identity across three decades, agree in sign in 114 of 120 snapshots, and correlate above 0.98 within each of the 20 runs, so the relation is not a between-condition scale effect. The realized gain is thus governed by the alignment of the endpoint difference with the evaluation-loss gradient, as

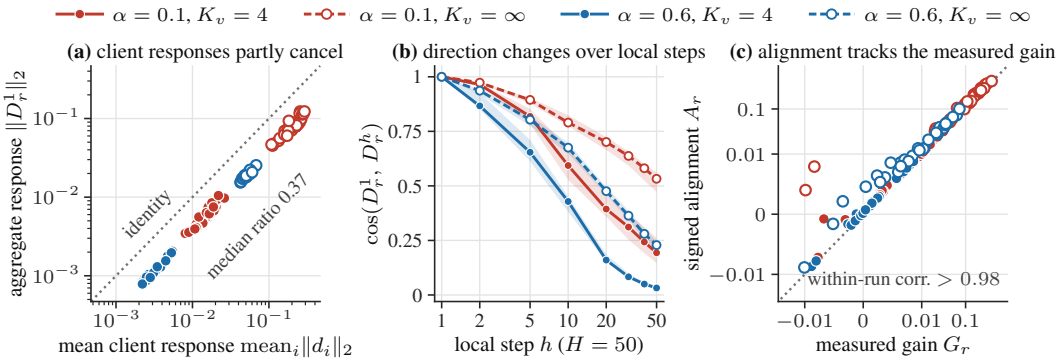

Figure 2: Adam-v transport diagnostics over 120 snapshots from 20 runs. Color distinguishes strong ($\alpha = 0.1$) and mild ($\alpha = 0.6$) data heterogeneity, and filled/solid versus hollow/dashed markers distinguish $K_v = 4$ and $K_v = \infty$. **(a)** Mean client-level first-step response versus the aggregate response. Points below the identity indicate cancellation across clients, and the aggregate response is typically much smaller, with a median ratio of 0.37. **(b)** Directional similarity between the first-step response D_r^1 and the response after h local steps. Lines show the median for each setting and bands show the interquartile range. **(c)** Signed alignment A_r versus the measured transport gain G_r. Within every run, their Pearson correlation exceeds 0.98.

Proposition 2 states. Figure A1 in Appendix D contrasts this with the endpoint displacement, whose within-run correlation with the gain has no consistent sign.

Separating delivery from the correction schedule. Expanding K_M in the evaluated FedMuon configuration also applies the global-direction correction less often. With the correction kept active every round, the paired $K_M = 1 \to 2$ drop falls from 7.97 ± 2.97 to 0.82 ± 1.47 points over five seeds at $\alpha = 0.1$ (Appendix E).

Takeaway: For Adam v, client responses to transport partly cancel after aggregation, and the realized gain follows the signed alignment of the endpoint difference rather than its size. For the evaluated FedMuon policy, most of the sensitivity to K_M comes from the correction schedule.

6.3 Adaptive Synchronization Tracks the Fixed-Interval Behavior

Table 1: Adaptive optimizer-memory synchronization compared with every-round ($K_x = 1$) and fully local ($K_x = \infty$) synchronization. $\bar{K}_x$ is the final controller interval, averaged over seeds. Accuracy is final test accuracy (%). Uplink is GB/round summed over participating clients and averaged over training; adaptive uplink includes probe traffic.

			Accuracy (%)			Uplink (GB/round)		
Memory	α	$\bar{K}_x$	Adaptive	$K_x = 1$	$K_x = \infty$	Adaptive	$K_x = 1$	$K_x = \infty$
CIFAR-10 / ResNet-18, 10 clients, full participation								
Adam v	0.1	3.80	83.26±1.43	80.29±2.59	76.85±8.61	0.66	0.89	0.45
	0.6	22.40	90.59±0.35	90.24±0.52	90.06±0.77	0.57	0.89	0.45
Muon M	0.1	1.00	84.54±2.37	85.60±2.10	49.34±9.76	0.89	0.89	0.45
	0.6	2.60	92.28±0.63	92.31±0.28	92.65±0.28	0.79	0.89	0.45
CIFAR-100 / ResNet-18, 100 clients, $q = 10\%$								
Adam v	0.1	256.00	62.22±0.15	62.90±0.69	61.13±0.61	0.47	0.90	0.45
	0.6	256.00	65.47±0.27	65.42±0.40	64.58±0.42	0.50	0.90	0.45
Muon M	0.1	1.00	57.27±1.04	56.91±1.09	28.40±2.90	0.90	0.90	0.45
	0.6	21.70	66.14±0.57	66.46±0.07	65.76±0.65	0.86	0.90	0.45
CIFAR-100 / ViT-Tiny, 100 clients, $q = 10\%$								
Adam v	0.1	256.00	42.63±1.06	42.73±0.98	42.46±1.20	0.22	0.43	0.22
	0.6	256.00	43.77±0.74	43.55±0.90	43.23±0.33	0.24	0.43	0.22
Muon M	0.1	256.00	40.75±1.09	41.00±0.37	36.30±0.65	0.23	0.43	0.22
	0.6	256.00	44.13±0.60	45.31±0.46	43.16±0.94	0.23	0.43	0.22

If the probe is a useful signal, the controller should hold $K_x = 1$ where Section 6.1 found expansion costly, and lengthen K_x where it was harmless. For Muon M on CIFAR-10 at $\alpha = 0.1$, where $K_M = 2$ already costs 33 points, every probe should return a positive gain; for Adam v on CIFAR-100, where every finite K_v stays within 0.6 points of $K_v = 1$, probes should return negative gains and K_v should keep doubling. Table 1 and Figure 1(c) show that this is what happens on ResNet-18, and Appendix F gives the remaining trajectories.

Adam v on ResNet-18. On CIFAR-10, the final interval expands from $\bar{K}_v = 3.8$ ($\alpha = 0.1$) to 22.4 ($\alpha = 0.6$), matching or exceeding every-round accuracy while reducing uplink by 25.8% and 36.0%. On CIFAR-100, the controller doubles the interval to $K_v = 256$, the largest value reachable within 300 rounds, cutting uplink by 44–48% (nearing fully local cost). This incurs no accuracy loss at $\alpha = 0.6$ and only a 0.68-point drop at $\alpha = 0.1$, where it still outperforms local training.

Muon M on ResNet-18. Under strong heterogeneity, every adaptive run maintains $K_M = 1$, matching the steep policy degradation of FedMuon beyond one round while remaining within one standard deviation of full synchronization. Under mild heterogeneity, the controller relaxes the interval ($\bar{K}_M = 2.6$ on CIFAR-10; $\bar{K}_M = 21.7$ late in training on CIFAR-100) at comparable accuracy, trimming uplink by 11.2% and 4.4%.

Because the synchronized memory has one entry per parameter, these savings grow with model size. A 7B-parameter model in 32-bit precision uploads 56 GB per client per round under full synchronization, so the 48% reduction for Adam v would save about 27 GB per client per round.

Transfer to ViT-Tiny. Adam v mirrors its ResNet-18 behavior; the controller expands K_v to 256, maintaining $K_v = 1$ accuracy while reducing uplink by 44–49%. For Muon M, the controller relaxes synchronization at both heterogeneity levels, cutting uplink by $\sim 46\%$. At $\alpha = 0.1$, performance matches $K_M = 1$ (40.75% vs. 41.00%) and exceeds local training by 4.45 points. At $\alpha = 0.6$, every-round synchronization holds a 1.2-point advantage (45.31% vs. 44.13%). After the first probe kept $K_M = 1$, subsequent gains were negative and extended the interval, an instance of the gap between snapshot federability and policy utility (Section 2).

Takeaway: Without task-specific tuning, the controller recovers fixed-interval sensitivities on ResNet-18, reducing net uplink by up to 48% while staying within ~ 1 point of full synchronization. On ViT-Tiny, its relaxation of Muon M under mild heterogeneity costs 1.2 points, illustrating how short-horizon probes can miss multi-round cumulative benefits.

7 Related Work

Beyond model averaging in FedAvg (McMahan et al., 2017), Mime broadcasts server-side optimizer states (Suresh et al., 2020), FedAdamW communicates block-mean second-moments (Liu et al., 2026), and FedMuon aggregates client momentum (Liu et al., 2025); each fixes the schedule on which the shared state is synchronized. DES-LOC decouples the clocks of optimizer states and parameters in data-parallel training (Iacob et al., 2026). In FL, heterogeneous data and intermittent participation make local optimizer states client-specific before aggregation, so the appropriate synchronization frequency depends on the setting. We isolate this timescale for conventional optimizer memories and adapt it online with a probe. Appendix H discusses further related work.

8 Conclusion

We studied optimizer-memory federation using a synchronization clock decoupled from model communication, distinguishing snapshot federability from long-run policy utility. For Adam v, the aggregate transport response depends on signed endpoint alignment rather than state discrepancy alone. For Muon M, synchronization sensitivity is strongly tied to the correction schedule coupled to momentum delivery. Our transport-gain controller adapts synchronization online using realized loss contrast, recovering fixed-interval sensitivities and cutting uplink by up to 48% while remaining within ~ 1 point of full synchronization. Our study remains limited to image classification on ResNet-18 and ViT-Tiny, and the underlying heuristic uses only gain signs, which may miss multi-round cumulative benefits. Extending the method to larger architectures and language tasks, alongside designing policy-aware controllers, remains key future work.

AI USE STATEMENT

In this work, we used generative AI tools only for language editing, including improving the clarity, grammar, and phrasing of the manuscript. All AI-assisted edits were reviewed by the authors. We take full responsibility for the final content of this work.

ETHICS STATEMENT

This work studies optimization algorithms for federated learning on standard public benchmarks (CIFAR-10 and CIFAR-100). It involves no human subjects, private data collection, or application-specific deployment, and we foresee no direct negative ethical or social impact.

REPRODUCIBILITY STATEMENT

Algorithm 1 and Section 4 specify the probe and the controller, and Appendix B contains all proofs. Section 5 and Appendix C give the datasets, partitions, participation, rounds, local steps, and seeds behind every table and figure. The anonymized code implementing the probe, the controller, and all baselines is available at `https://anonymous.4open.science/r/adaptive-federated-optimizer-anon-84B2`.

REFERENCES

Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Ali Behrouz, Meisam Razaviyayn, Peilin Zhong, and Vahab Mirrokni. Nested learning: The illusion of deep learning architectures. *Advances in Neural Information Processing Systems*, 38:46968–47002, 2025.

Hong Chen, Pengcheng Wu, Yuanguo Lin, Peilin Zhao, Xiuze Zhou, Fan Lin, and Han Yu. Federated Nested Learning: Collaborative training of self-referential memories for test-time adaptation. *arXiv preprint arXiv:2605.16350*, 2026.

Xiumei Deng, Jun Li, Kang Wei, Long Shi, Zehui Xiong, Ming Ding, Wen Chen, Shi Jin, and H Vincent Poor. Towards communication-efficient federated learning via sparse and aligned adaptive optimization. *IEEE Transactions on Signal Processing*, 2025.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Alex Iacob, Lorenzo Sani, Mher Safaryan, Paris Giampouras, Samuel Horváth, Andrej Jovanovic, Meghdad Kurmanji, Preslav Aleksandrov, William Shen, Xinchi Qiu, et al. DES-LOC: Desynced low communication adaptive optimizers for foundation models. In *International Conference on Learning Representations*, pp. 42813–42866, 2026.

Divyansh Jhunjhunwala, Pranay Sharma, Aushim Nagarkatti, and Gauri Joshi. Fedvarp: Tackling the variance due to partial client participation in federated learning. In *Uncertainty in Artificial Intelligence*, pp. 906–916. PMLR, 2022.

Keller Jordan, Yuchen Jin, Vlado Boza, You Jiacheng, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL `https://kellerjordan.github.io/posts/muon/`.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pp. 5132–5143. PMLR, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alexander Kolesnikov, Alexey Dosovitskiy, Dirk Weissenborn, Georg Heigold, Jakob Uszkoreit, Lucas Beyer, Matthias Minderer, Mostafa Dehghani, Neil Houlsby, Sylvain Gelly, Thomas Unterthiner, and Xiaohua Zhai. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Sunwoo Lee, Tuo Zhang, and A. Salman Avestimehr. Layer-wise adaptive model aggregation for scalable federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37-7, pp. 8491–8499, 2023.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020.

Junkang Liu, Fanhua Shang, Junchao Zhou, Hongying Liu, Yuanyuan Liu, and Jin Liu. FedMuon: Accelerating federated learning with matrix orthogonalization. *arXiv preprint arXiv:2510.27403*, 2025.

Junkang Liu, Fanhua Shang, Hongying Liu, Yuxuan Tian, Yuanyuan Liu, Jin Liu, Kewen Zhu, and Zhouchen Lin. FedAdamW: A communication-efficient optimizer with convergence and generalization guarantees for federated large models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 23748–23756, 2026.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pp. 1273–1282. PMLR, 2017.

Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and H Brendan McMahan. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.

Amirhossein Reisizadeh, Aryan Mokhtari, Hamed Hassani, Ali Jadbabaie, and Ramtin Pedarsani. FedPAQ: A communication-efficient federated learning method with periodic averaging and quantization. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your ViT? Data, augmentation, and regularization in vision transformers. *arXiv preprint arXiv:2106.10270*, 2021.

Sebastian U. Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified SGD with memory. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Ananda Theertha Suresh, Martin Jaggi, Mehryar Mohri, Sai Praneeth Karimireddy, Sashank Reddi, Satyen Kale, and Sebastian Stich. Mime: Mimicking centralized stochastic algorithms in federated learning. In *ICML 2021*, 2020.

Yuki Takezawa, Anastasia Koloskova, Xiaowen Jiang, and Sebastian Stich. Fedmuon: Federated learning with bias-corrected LMO-based optimization. In *International Conference on Learning Representations*, volume 2026, pp. 39538–39569, 2026.

Feijie Wu, Song Guo, Zhihao Qu, Shiqi He, Ziming Liu, and Jing Gao. Anchor sampling for federated learning with partial client participation. In *International conference on machine learning*, pp. 37379–37416. PMLR, 2023.

Chenguang Xiao and Shuo Wang. Rethinking the initialization of momentum in federated learning with heterogeneous data. *arXiv preprint arXiv:2411.19798*, 2024.

Jing Xu, Sen Wang, Liwei Wang, and Andrew Chi-Chih Yao. FedCM: Federated learning with client-level momentum. *arXiv preprint arXiv:2106.10874*, 2021.

A Background on the Optimizers

We review the optimizers whose states we federate, adopting the notation of the main text. Throughout, client j performs local steps $h = 1, \dots, H$ in round r starting from the server model w^r, where $g^r_{j,h}$ denotes its stochastic gradient at step h. The model parameters are averaged using FedAvg (McMahan et al., 2017) every round, while the optimizer memories follow independent synchronization intervals.

Adam. Adam (Kingma & Ba, 2014) maintains two exponential moving averages of the stochastic gradient: the first moment m and the elementwise second moment v, defined as

$$m^r_{j,h} = \beta_1 m^r_{j,h-1} + (1-\beta_1) g^r_{j,h}, \qquad v^r_{j,h} = \beta_2 v^r_{j,h-1} + (1-\beta_2)\big(g^r_{j,h}\big)^{\odot 2}. \tag{A1}$$

Let t denote the client's Adam step counter, $\hat{m} = m/(1-\beta_1^t)$ and $\hat{v} = v/(1-\beta_2^t)$ denote the bias-corrected moments, η be the learning rate, and λ be the decoupled weight decay factor as in AdamW. The parameter update rule is given by

$$w^r_{j,h} = w^r_{j,h-1} - \eta \left(\frac{\hat{m}^r_{j,h}}{\sqrt{\hat{v}^r_{j,h}} + \epsilon} + \lambda\, w^r_{j,h-1} \right), \tag{A2}$$

where the square root and division are evaluated element-wise. AdamW (Loshchilov & Hutter, 2019) differs from original Adam solely in its application of weight decay. Standard Adam adds λw directly to the gradient, causing weight decay to be rescaled by $1/(\sqrt{\hat{v}} + \epsilon)$ alongside the gradient; AdamW instead subtracts $\eta\lambda w$ directly from the weights. The two formulations coincide when $\lambda = 0$. Consequently, the second moment v acts as a per-coordinate adaptive step size, meaning that replacing v fundamentally alters how strongly each coordinate responds to new gradient information. Proposition 1 provides an exact characterization of this effect for the first step following transport. In our setup, m is reset at the beginning of each local segment, and only v is federated.

FedAdamW. FedAdamW (Liu et al., 2026) federates Adam's second-moment information in a compressed format. Instead of transmitting the full element-wise state v, clients transmit a block-mean summary containing a single scalar per parameter block, thereby keeping communication overhead low. By contrast, we federate the full coordinate-level state v, making memory synchronization a significant communication investment whose interval warrants adaptive control. Appendix G provides a detailed comparison between these two representations under our proposed controller.

Muon. Muon (Jordan et al., 2024) operates on each weight matrix W as an indivisible unit. It accumulates the matrix gradient $G^r_{j,h}$ into a momentum buffer, normalizes the buffer, and applies several Newton–Schulz iterations to obtain an approximate orthogonalization:

$$M^r_{j,h} = \mu M^r_{j,h-1} + G^r_{j,h}, \quad O^r_{j,h} = \mathrm{NS}\left(\frac{M^r_{j,h}}{\|M^r_{j,h}\|_F + \epsilon_{\mathrm{N}}} \right), \quad W^r_{j,h} = W^r_{j,h-1} - \eta\, O^r_{j,h}, \tag{A3}$$

where $\epsilon_{\mathrm{N}} > 0$ is a small numerical stabilization constant. Let $M = U\Sigma V^\top$ be the singular value decomposition of M; the operator NS approximates the orthogonal polar factor $UV^\top$. The update thereby preserves the singular directions of the momentum buffer while equalizing their singular values. Parameters with fewer than two dimensions default to standard momentum updates. The momentum buffer M serves as the federated memory in our setting and is aggregated prior to orthogonalization, as an average of orthogonal matrices is generally not orthogonal.

FedMuon. FedMuon (Liu et al., 2025) federates Muon's momentum buffer and incorporates a local–global directional correction. On synchronization rounds, clients reinitialize their momentum from the aggregated matrix $\bar{M}^r$, and their local update direction is blended with a global direction

derived from aggregated model updates using the weighting specified in Table A1. In our evaluation setup, this correction step occurs exclusively on delivery rounds, meaning the momentum interval K_M simultaneously governs the frequency of this directional correction. Section 6.2 isolates and analyzes these two conflated mechanisms independently.

B Proofs of the Transport-Response Results

Adam first-step conditions. Fix a nonempty participating set $\mathcal{S}_r$ at a segment boundary r. Let the incoming raw second moments be finite vectors $v_i^r \in [0,\infty)^p$ for some dimension p, and let $\bar{v}^r$ be their equal-weight mean over the participating clients as in Section 4.1. For each client, the local and transport branches start from the same model w^r, use the same first minibatch and gradient $g_i \in \mathbb{R}^p$, and start from the same incoming first-moment m_i^{in}. They differ only in the incoming second-moment, which is v_i^r in the local branch and $\bar{v}^r$ in the transport branch. Both use the real-arithmetic AdamW recurrence in minimization mode with no AMSGrad, zero weight decay, and shared hyperparameters satisfying $\eta \geq 0$, $0 \leq \beta_1, \beta_2 < 1$, and $\epsilon > 0$.

Each client carries its own Adam step counter across segments, advanced by that client's completed local segments. We write $t_i \geq 1$ for client i's counter after the first-step increment, which is the index used in both bias corrections. Both probe branches of a client inherit the same t_i. In our experiments, the first moment is not federated and is reset to zero at the start of each local segment, so $m_i^{\text{in}} = 0$. Define

$$\begin{aligned} m_i &= \beta_1 m_i^{\text{in}} + (1-\beta_1) g_i, \quad c_i = \sqrt{1-\beta_2^{t_i}}, \\ a_i &= \beta_2 \bar{v}^r + (1-\beta_2) g_i^{\odot 2}, \quad b_i = \beta_2 v_i^r + (1-\beta_2) g_i^{\odot 2}, \end{aligned} \tag{A4}$$

and define the Adam response coefficient coordinatewise by

$$s_{ik} = \begin{cases} \dfrac{\eta \beta_2 m_{ik}}{(1-\beta_1^{t_i}) c_i \big(\sqrt{a_{ik}} + \sqrt{b_{ik}}\big)\big(\sqrt{a_{ik}}/c_i + \epsilon\big)\big(\sqrt{b_{ik}}/c_i + \epsilon\big)}, & a_{ik} + b_{ik} > 0, \\ 0, & a_{ik} = b_{ik} = 0. \end{cases} \tag{A5}$$

Proof of Proposition 1. The two branches enter the first step with the same first-moment and gradient, so their common updated first-moment is m_i from Eq. (A4), with bias-corrected value $m_i/(1-\beta_1^{t_i})$. The updated raw second-moments are a_i and b_i, and the two first-step increments are

$$\Delta w_i^{\text{transport},1} = -\frac{\eta}{1-\beta_1^{t_i}} \frac{m_i}{\sqrt{a_i}/c_i + \epsilon}, \qquad \Delta w_i^{\text{local},1} = -\frac{\eta}{1-\beta_1^{t_i}} \frac{m_i}{\sqrt{b_i}/c_i + \epsilon}, \tag{A6}$$

with quotients and square roots taken coordinatewise and with ϵ outside the square root of the corrected second-moment.

Fix a coordinate k with $a_{ik} + b_{ik} > 0$ and write $A = \sqrt{a_{ik}}/c_i + \epsilon$ and $B = \sqrt{b_{ik}}/c_i + \epsilon$. Then

$$\frac{1}{A} - \frac{1}{B} = \frac{b_{ik} - a_{ik}}{c_i\big(\sqrt{a_{ik}} + \sqrt{b_{ik}}\big) AB}. \tag{A7}$$

Since $b_{ik} - a_{ik} = \beta_2(v_{ik}^r - \bar{v}_k^r)$, subtracting the two signed increments in Eq. (A6) gives $d_{ik} = s_{ik}(\bar{v}_k^r - v_{ik}^r)$ with s_{ik} from Eq. (A5).

If $a_{ik} = b_{ik} = 0$, both denominators in Eq. (A6) equal ϵ and the numerators are common, so $d_{ik} = 0$, consistent with the definition of s_{ik}. If only one of a_{ik} and b_{ik} is zero, the same rationalization applies because their square-root sum is positive. The identity also includes equal incoming memories, $\beta_2 = 0$, $\eta = 0$, and $\beta_1 = 0$ without division by $\bar{v}_k^r - v_{ik}^r$.

Finally, the same participating clients and equal weights define both $\bar{v}^r$ and D_r^1, so $|\mathcal{S}_r|^{-1} \sum_i (\bar{v}^r - v_i^r) = 0$. Subtracting the mean coefficient $\bar{s}$ therefore leaves the aggregate unchanged and gives Eq. (6). □

A norm bound. For reference, define

$$Q_v = \frac{1}{|\mathcal{S}_r|} \sum_{i \in \mathcal{S}_r} \|v_i^r - \bar{v}^r\|_2^2, \qquad V_s = \frac{1}{|\mathcal{S}_r|} \sum_{i \in \mathcal{S}_r} \|s_i - \bar{s}\|_2^2. \tag{A8}$$

The aggregate identity implies

$$\|D_r^1\|_2 \le \sqrt{Q_v V_s}. \tag{A9}$$

Indeed, $\|u \odot z\|_2 \le \|u\|_2 \|z\|_2$ for real vectors, so the triangle inequality followed by Cauchy–Schwarz over clients gives

$$\|D_r^1\|_2 \le \frac{1}{|\mathcal{S}_r|} \sum_i \|s_i - \bar{s}\|_2 \|\bar{v}^r - v_i^r\|_2 \le \sqrt{V_s Q_v}. \tag{A10}$$

Same memories, different aggregate response. Take two clients and one coordinate with $v_1^r = 0$, $v_2^r = u > 0$, $\eta > 0$, and $0 < \beta_2 < 1$. Then $\bar{v}^r = u/2$ and $Q_v = u^2/4$. If both current gradients vanish, $s_1 = s_2 = 0$ and $D_r^1 = 0$. Keeping the same incoming memories but taking $g_1 = h > 0$ and $g_2 = -h$ gives $s_1 > 0$ and $s_2 < 0$ by Eq. (A5), hence $D_r^1 = \frac{u}{4}(s_1 - s_2) > 0$. Thus the same incoming memory dispersion can yield different immediate aggregate responses. A rule that acts on memory discrepancy alone would treat the two configurations identically, although transport changes the aggregate update in one and not in the other.

Proof of Proposition 2. Write $F = L_{\text{eval}}$, $y = w_{\text{local}}^{r,H}$, and $D = D_r^H$. By Eq. (7), $y + D = w^r + \overline{\Delta w}_{\text{transport}}^{r,H}$, so $G_r = F(y) - F(y + D)$. The case $D = 0$ is immediate with $R_r = 0$.

For $D \neq 0$, the hypothesis of the proposition reads

$$\|\nabla F(y + uD) - \nabla F(y)\|_2 \le Lu\|D\|_2, \qquad 0 \le u \le 1. \tag{A11}$$

Define $\phi(u) = F(y + uD)$ on $[0, 1]$. Along the segment,

$$\phi'(u) = \langle \nabla F(y + uD), D \rangle. \tag{A12}$$

By Condition (A11) and Cauchy–Schwarz, $|\phi'(u)| \le |\langle \nabla F(y), D \rangle| + L\|D\|_2^2$ on $(0, 1)$. Hence the mean value theorem makes ϕ Lipschitz and therefore absolutely continuous, so the fundamental theorem of calculus gives

$$F(y + D) - F(y) = \langle \nabla F(y), D \rangle + \int_0^1 \langle \nabla F(y + uD) - \nabla F(y), D \rangle \, du. \tag{A13}$$

Define

$$R_r = -\int_0^1 \langle \nabla F(y + uD) - \nabla F(y), D \rangle \, du. \tag{A14}$$

Then $G_r = -\langle \nabla F(y), D \rangle + R_r$. By Cauchy–Schwarz and Condition (A11),

$$\begin{aligned} |R_r| &\le \int_0^1 \|\nabla F(y + uD) - \nabla F(y)\|_2 \|D\|_2 \, du \\ &\le \int_0^1 Lu\|D\|_2^2 \, du = \frac{L}{2}\|D\|_2^2. \end{aligned} \tag{A15}$$

□

Equal endpoint distances can yield opposite gains. For $F(z) = z^2/2$ and a local endpoint $x = a > 0$, consider $D_- = -h$ and $D_+ = h$ with $0 < h < 2a$. Both have distance h from the local endpoint, but

$$G(D_-) = ah - \frac{h^2}{2} > 0, \qquad G(D_+) = -ah - \frac{h^2}{2} < 0. \tag{A16}$$

Thus endpoint distance alone does not determine the sign of the measured gain.

C Additional Experimental Details

Datasets and partitioning. For CIFAR-10 and CIFAR-100, we first remove a class-balanced 10% of the training set as a server-held validation split, which is used only to evaluate L_{eval} in the transport-gain probe. We partition the remaining training data across clients with a Dirichlet distribution over labels with concentration $\alpha \in \{0.1, 0.6\}$. We report accuracy on the standard test set. CIFAR-10

uses 10 clients with full participation for 100 rounds, and CIFAR-100 uses 100 clients with 10 clients sampled uniformly without replacement per round for 300 rounds. Every round consists of 50 local optimizer steps per participating client. For all vision benchmarks, we augment training images with random 32×32 crops after 4-pixel padding and random horizontal flips.

Optimizer memories and synchronization. For Adam, the first moment m is reset to zero at the start of each local segment and is never federated, and the full-coordinate second moment v is aggregated with equal weights over participating clients at synchronization rounds. Each client carries its own Adam step counter across segments. For Muon, synchronization delivers the equal-weight mean of participating clients' momentum, aggregated before orthogonalization, and the FedMuon global-direction correction is applied on the same rounds. Between deliveries, clients train with their local momentum. Newton–Schulz orthogonalization is applied to every parameter with at least two dimensions. One-dimensional parameters and the classifier head instead take a plain momentum step from the same momentum buffer.

Optimizer hyperparameters. Table A1 lists the hyperparameters of each benchmark. They are shared by all synchronization intervals and by the adaptive runs within a benchmark, so the runs differ only in when optimizer memory is synchronized.

Table A1: Optimizer and training hyperparameters.

	CIFAR-10 ResNet-18	CIFAR-100 ResNet-18	CIFAR-100 ViT-Tiny
Local batch size	50	50	50
Adam			
Learning rate	1e−3	1e−3	3e−4
(β_1, β_2)	$(0.9, 0.999)$	$(0.9, 0.999)$	$(0.9, 0.999)$
ϵ	1e−8	1e−8	1e−8
Weight decay	0	0	0.01
Muon / FedMuon			
Learning rate	1e−2	1e−2	3e−2
Momentum coefficient	0.95	0.95	0.95
Newton–Schulz iterations	5	5	5
Weight decay	0	0	0.01
Global-direction blending weight	0.5	0.5	0.5
Learning-rate schedule	None	None	Cosine

Probe and controller. Each probe runs both branches for $H = 50$ local steps, the full local segment, from the same model, minibatch sequence, and non-probed optimizer states. The server evaluates L_{eval} at both aggregated branch endpoints on the validation split, and the controller follows Algorithm 1 from $K_x = 1$ and $J_x = 1$. The transport diagnostics of Section 6.2 use the same branch construction but evaluate endpoint losses on a fixed set of 2048 training examples under evaluation transforms, and they are recorded without altering the training trajectory.

Probing cost. Each probing round adds two H-step branch runs per participating client and two evaluations of L_{eval} at the server, so relative to training without probing, the controller adds client computation equal to $2P/R$, where P is the number of probing rounds over R rounds. Because the probing interval backs off exponentially, P is small. Across our adaptive runs, P ranges from 7 to 13 over the 100 rounds of CIFAR-10 and from 8 to 16 over the 300 rounds of CIFAR-100, corresponding to 14.0–26.0% and 5.3–10.7% additional client computation, respectively. The probe branches do not alter the training trajectory, so probing itself does not change per-round convergence. Its effect on training comes only through the selected intervals, which Table 1 and Appendix F report.

Comparisons. We compare each adaptive run against two fixed-interval references: synchronization every round ($K_x = 1$) and fully local optimizer memory ($K_x = \infty$).

Simulation and communication accounting. We simulate all clients within a single training process and execute client sampling, local optimization, and server aggregation sequentially rather than over a network. We therefore compute communication analytically from the size of the transferred tensors, using 32-bit values. Uplink per round sums, over participating clients, the model update, the optimizer memory on synchronization rounds, and both branch updates on probe rounds.

Seeds. Each seed controls both the Dirichlet client-data partition and the local training randomness, including minibatch order, dropout, and the initialization of newly added classification heads. We use five seeds for CIFAR-10 and three for CIFAR-100.

Compute and software. All experiments were run on an internal SLURM-managed cluster using a mix of NVIDIA RTX 4090, RTX A6000, RTX PRO 6000, RTX 6000 Ada, RTX 3090, and A100 GPUs, with each training run assigned a single GPU. We implement all methods in PyTorch (versions 2.13-2.14 across cluster nodes). All experiments run in full precision.

D Additional Evidence for the Adam Transport-Gain Analysis

This appendix provides detailed evidence for the Adam transport-gain analysis summarized in Figure 2. We examine three links in that analysis. First, we verify the exact first-step response of Proposition 1 and quantify how client-level responses combine after aggregation. Second, we track the first-step difference through the full local segment to connect D_r^1 to the realized endpoint difference D_r^H. Third, we evaluate the loss-space decomposition of Proposition 2 on the realized endpoints. All results use the 120 diagnostic snapshots described in Section 5, six per run over 20 runs, and all correlations are Pearson correlations.

D.1 First-Step Aggregate Response

We first verify that Proposition 1 describes the implemented first-step response and then examine what determines the resulting aggregate response. The closed-form expression reproduces the implemented update with a relative residual between 1.1×10^{-5} and 1.9×10^{-3} across all snapshots, median 1.3×10^{-4}, with no degenerate case. Table A2 reports the response by condition. The absolute response scale differs by two orders of magnitude across conditions, whereas the ratio of the aggregate response $\|D_r^1\|_2$ to the mean client response $\frac{1}{|\mathcal{S}_r|}\sum_i \|d_i\|_2$ stays within 0.36–0.40.

Table A2: First matched step of Adam-v transport, by condition. Entries are the mean over five seeds of the per-run mean, with the sample standard deviation across seeds. CIFAR-10, ResNet-18, $H = 50$.

α	K_v	mean client response	aggregate response	ratio
0.1	4	0.01643 ± 0.00282	0.006529 ± 0.00129	0.3963 ± 0.0148
0.1	∞	0.20149 ± 0.03990	0.080163 ± 0.02130	0.3990 ± 0.0294
0.6	4	0.003330 ± 0.000257	0.001205 ± 0.000101	0.3611 ± 0.0031
0.6	∞	0.05132 ± 0.00330	0.019037 ± 0.00140	0.3711 ± 0.0057

To test the central implication of Proposition 1, we ask whether a scalar measure of memory dispersion alone explains the aggregate response. We use the dispersion statistic of Eq. (A8), $Q_v = \frac{1}{|\mathcal{S}_r|}\sum_{i\in\mathcal{S}_r} \|v_i^r - \bar{v}^r\|_2^2$, the mean squared deviation of the incoming client memories from their aggregate. Pooled over all 120 snapshots, Q_v and $\|D_r^1\|_2$ correlate at $+0.78$, but this association is driven largely by the scale differences between the four (α, K_v) conditions. Within a condition, the correlation ranges from -0.00 to $+0.68$ ($+0.54$, $+0.29$, $+0.68$, and -0.00 for $(0.1, 4)$, $(0.1, \infty)$, $(0.6, 4)$, and $(0.6, \infty)$), so dispersion alone does not consistently determine the aggregate response. This is consistent with Eq. (6), in which the aggregate depends on how the client-specific response coefficients covary with the memory deviations rather than on the size of those deviations.

D.2 From the First Step to the Realized Endpoint

Proposition 1 is exact for the first matched step, whereas the probe follows both branches for $H = 50$ local steps. We therefore track the induced difference across the local segment to determine how the first-step response evolves into the realized endpoint difference D_r^H. Table A3 shows that the direction turns steadily while the magnitude grows, with the median cosine falling to 0.219 and the median norm ratio reaching 62.2 at $h = 50$. The endpoint difference that enters Proposition 2 is therefore taken from the realized trajectories rather than extrapolated from the first step.

Table A3: Evolution of the transport-induced difference over the local segment, as the median and interquartile range over 120 snapshots. The norm ratio is heavy-tailed, so quantiles are reported.

local step h	$\cos(D_r^1, D_r^h)$	$\|D_r^h\|_2/\|D_r^1\|_2$
2	0.944 [0.920, 0.969]	2.45 [2.29, 2.54]
5	0.807 [0.744, 0.867]	8.04 [6.27, 8.93]
10	0.653 [0.492, 0.747]	17.8 [12.5, 21.7]
20	0.460 [0.263, 0.604]	31.4 [23.7, 49.7]
30	0.349 [0.156, 0.501]	40.4 [29.4, 71.4]
40	0.268 [0.103, 0.422]	50.3 [33.9, 94.4]
50	0.219 [0.076, 0.349]	62.2 [36.7, 116.0]

D.3 Loss-Space Effect of the Realized Endpoint Difference

We next evaluate the loss-space decomposition of Proposition 2. If the first-order alignment term A_r captures the dominant local effect of the realized transport difference, it should closely track the directly measured gain G_r within each run. It does: A_r and G_r correlate at $+0.9961$ pooled over all snapshots and above $+0.98$ within each of the 20 runs, they agree in sign in 114 of 120 snapshots, and the relative remainder $|R_r|/|G_r|$ has median 0.22 and interquartile range $[0.11, 0.39]$ over the 118 snapshots with $|G_r| > 10^{-4}$. For reference, Figure A1 also reports the within-run association between the endpoint displacement $\|D_r^{50}\|_2$ and G_r. Pooled, the two correlate at $+0.68$, which is a between-condition effect, and within runs the correlation is negative in 12 of the 20 runs.

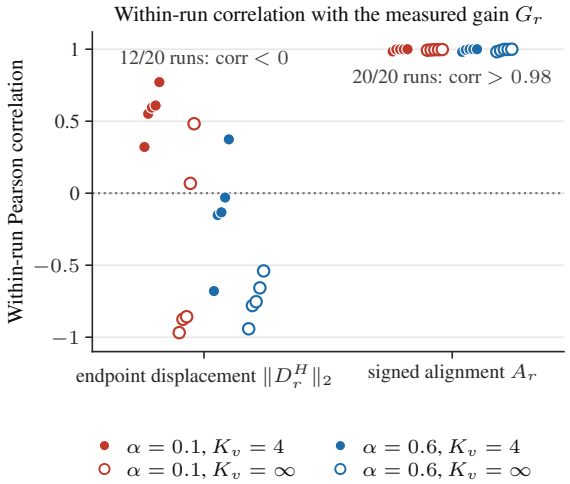

Figure A1: Within-run correlation with the measured gain G_r. Each point is one run and shows the Pearson correlation over that run's six snapshots. Left, endpoint displacement $\|D_r^H\|_2$ against G_r. Right, signed alignment A_r against G_r. Displacement is inconsistent within runs, with negative correlation in 12 of 20 runs, whereas alignment is consistently strong, exceeding 0.98 in all 20 runs. Color and marker fill follow Figure 2.

E FedMuon Correction-Schedule Ablation

The fixed-K_M sweep of Section 6.1 couples the frequency of momentum delivery to the correction schedule, since the evaluated configuration applies the global-direction correction on delivery rounds. This ablation separates the two by holding the correction active every round while varying the same delivery intervals. Table A4 reports both schedules. Each cell is five paired seeds on CIFAR-10 with ResNet-18 at $\alpha = 0.1$, with the blending weight, the delivery intervals, and every other setting fixed between the two schedules. The $K_M = 1$ row is the matched control, since momentum is delivered every round there and both schedules apply the correction on 99% of rounds. The two schedules agree there and separate once delivery becomes sparse.

Read down the two accuracy columns, expanding K_M from 1 to 2 costs 7.97 ± 2.97 points under the gated schedule and 0.82 ± 1.47 points under the always-on schedule over the same seed pairs. The

Table A4: FedMuon correction schedule against momentum delivery interval. Final accuracy (%) as mean and sample standard deviation over five paired seeds, with the paired difference between the schedules in the last column.

K_M	gated correction rate	gated on delivery	applied every round	always-on − gated
1	99% of rounds	83.93 ± 4.04	84.31 ± 3.70	$+0.38 \pm 1.04$
2	49% of rounds	75.96 ± 5.49	83.49 ± 4.92	$+7.53 \pm 3.02$
∞	never	73.86 ± 11.09	81.40 ± 4.58	$+7.54 \pm 7.69$

correction schedule therefore accounts for most of the sensitivity attributed to the delivery interval in the coupled policy.

F Training Dynamics of Adaptive Runs

This section shows how the adaptive runs in Table 1 evolve during training, including the synchronization interval selected by the controller (Appendix F.1) and the resulting training and test loss (Appendix F.2).

F.1 Interval Trajectories

Figure A2 shows how the controller's interval evolves during training. Because every change of K_x resets the probing interval to K_x (Eq. (9)), consecutive negative gains produce a staircase whose steps lengthen geometrically, while a flat line at $K_x = 1$ means that every measured gain was positive.

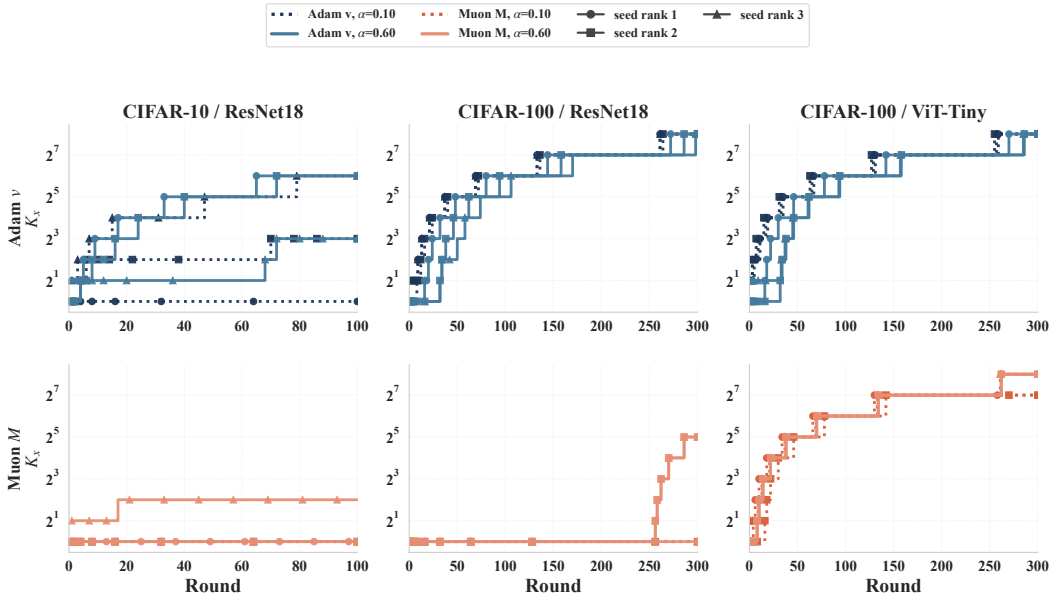

Figure A2: Synchronization interval K_x selected by the transport-gain controller over training, for Adam v (top) and Muon M (bottom). Each line is one seed, three per setting (for CIFAR-10, three of the five seeds are shown), with solid, dashed, and dotted lines for seed ranks 1–3. Color denotes heterogeneity ($\alpha = 0.1$ dark, $\alpha = 0.6$ light). The vertical axis is logarithmic.

For Adam v on CIFAR-10, the interval grows quickly at both heterogeneity levels but plateaus at small values under $\alpha = 0.1$, whereas it continues to grow under $\alpha = 0.6$. This matches the larger final interval at $\alpha = 0.6$ in Table 1. On CIFAR-100 with both architectures, the interval doubles at nearly every probing round and reaches its largest value by the end of training, consistent with the insensitivity of v to long intervals.

For Muon M on ResNet-18 under $\alpha = 0.1$, the interval stays at $K_M = 1$ throughout training on both datasets, since every measured gain is positive. Under $\alpha = 0.6$, the interval expands early and

stabilizes at small values on CIFAR-10. On CIFAR-100, it remains at $K_M = 1$ for most of training and expands only in the final rounds. This late expansion explains why the uplink reduction in this setting (4.4%) is small despite a final interval of $\bar{K}_M = 21.7$, since the final interval reflects the controller's end state rather than its average over training. On ViT-Tiny, by contrast, the interval expands at both heterogeneity levels, as discussed in Section 6.3.

F.2 Loss Trajectories

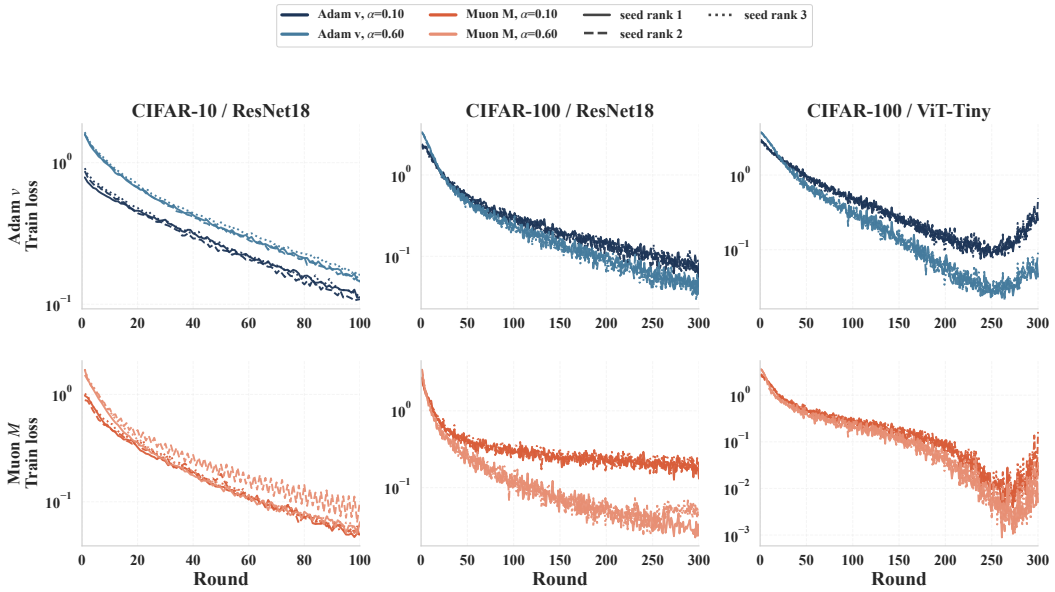

Figure A3: Training loss of adaptive runs for Adam v (top) and Muon M (bottom), measured as the mean local loss over participating clients. Color denotes heterogeneity ($\alpha = 0.1$ dark, $\alpha = 0.6$ light), and line style distinguishes seeds. For CIFAR-10, three of the five seeds are shown. The vertical axis is logarithmic.

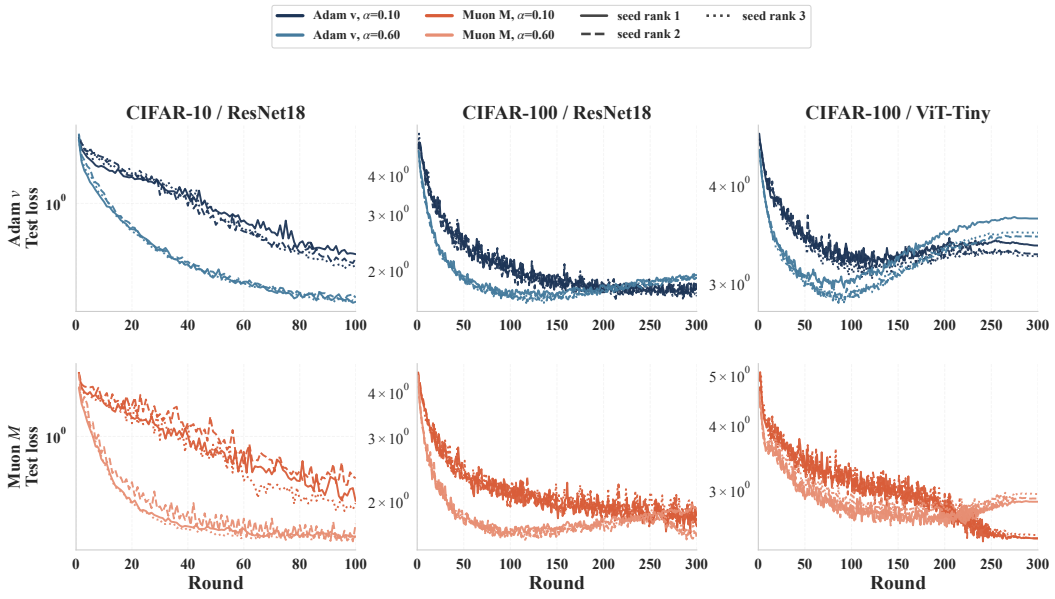

Figure A4: Test loss of the global model in adaptive runs, with the same layout as Figure A3.

Figures A3 and A4 show the training and test loss of the adaptive runs. Seeds follow closely matching trajectories in every setting, and no seed-specific instabilities appear under the controller's interval changes.

On ResNet-18, training loss decreases steadily for both memories. On CIFAR-10, test loss decreases throughout training and is consistently lower under mild heterogeneity. On CIFAR-100, test loss under $\alpha = 0.6$ reaches its minimum around rounds 100–150 and then rises slightly while training loss continues to fall, indicating late-stage overfitting. For Muon M on CIFAR-100 under $\alpha = 0.1$, where the controller holds $K_M = 1$, training loss plateaus early at a higher level than under $\alpha = 0.6$.

On ViT-Tiny, test loss under $\alpha = 0.6$ reaches its minimum near round 100 for both memories and increases afterwards, reflecting overfitting in later rounds. Under $\alpha = 0.1$, the test loss of Muon M remains high early in training and decreases mainly between rounds 150 and 220, as the controller expands its interval. Training loss on ViT-Tiny also becomes less stable toward the end of training.

G Full-Coordinate versus Block-Mean Second Moment

Our study federates Adam's full-coordinate second moment v, whereas FedAdamW communicates a block-mean representation with one scalar per parameter block (Liu et al., 2026). Table A5 compares the two under the same adaptive controller on CIFAR-10 with ResNet-18.

Table A5: Full-coordinate versus block-mean Adam v under adaptive synchronization on CIFAR-10 with ResNet-18 (10 clients, full participation, five seeds per cell), compared with every-round synchronization ($K_x = 1$). $\bar{K}_x$ is the final controller interval, averaged over seeds. Accuracy is final test accuracy (%). Uplink is GB/round summed over participating clients and averaged over training; adaptive uplink includes probe traffic.

			Accuracy (%)		Uplink (GB/round)	
v representation	α	$\bar{K}_x$	Adaptive	$K_x = 1$	Adaptive	$K_x = 1$
Full-coordinate	0.1	3.80	83.26±1.43	80.29±2.59	0.66	0.89
	0.6	22.40	90.59±0.35	90.24±0.52	0.57	0.89
Block-mean	0.1	7.60	80.47±2.76	77.18±2.92	0.45	0.45
	0.6	64.00	90.28±0.56	90.25±0.32	0.45	0.45

Under mild heterogeneity ($\alpha = 0.6$), the two representations perform comparably, with all four configurations yielding accuracies within 0.35 percentage points of one another. Under strong heterogeneity ($\alpha = 0.1$), the full-coordinate state improves accuracy by approximately 3 percentage points both under every-round synchronization (80.29% versus 77.18%) and under adaptive control (83.26% versus 80.47%). Averaging v within coarse blocks discards coordinate-wise variations across client memories, which become critical when client data distributions diverge strongly. For both representations, adaptive synchronization achieves higher mean accuracy than every-round synchronization, though under strong heterogeneity these gains remain within the margin of seed variation.

Because the block-mean state is already compact (Liu et al., 2026), lengthening its interval leaves uplink almost unchanged, even though the controller also selects long intervals for it ($\bar{K}_x = 7.6$ and 64). For the full-coordinate state, which doubles the per-round payload at $K_x = 1$, the controller removes 26–37% of that cost. Compression and interval adaptation are thus complementary, since compression reduces how much of an optimizer memory is sent and the interval determines when. The interval matters most for memories whose synchronization is expensive, such as full-coordinate v or Muon momentum, whose singular directions admit no comparably compact block summary.

H Extended Related Work

Federated optimization under heterogeneity. FedAvg established local updates with periodic model averaging (McMahan et al., 2017). Under heterogeneous client data, local updates drift toward client-specific optima, which FedProx mitigates with a proximal term and SCAFFOLD with control variates (Li et al., 2020; Karimireddy et al., 2020). Partial participation adds variance from client sampling, which FedVARP and FedAMD address (Jhunjhunwala et al., 2022; Wu et al., 2023).

Our analysis in Section 3 shows how these two factors, heterogeneity and participation, shape the optimizer memories that are later aggregated.

Adaptive optimizers and optimizer state in FL. FedOpt moves adaptive optimization to the server while clients run local SGD (Reddi et al., 2020), and Mime broadcasts server-side optimizer statistics so that local updates follow centralized optimization more closely (Suresh et al., 2020). Other work studies momentum under heterogeneity, including client-level momentum and momentum initialization (Xu et al., 2021; Xiao & Wang, 2024). More recent methods federate the local optimizer's state directly. FedAdamW communicates block-mean second-moment information (Liu et al., 2026), and FedMuon aggregates client momentum and aligns local updates with the global update direction (Liu et al., 2025). A separately proposed method, also named FedMuon, corrects the bias of Muon's linear minimization oracle when Muon is used as the local optimizer of FedAvg (Takezawa et al., 2026). These methods establish that optimizer state can benefit FL, but each fixes when the shared state is synchronized. We treat that schedule as a variable and study it separately for each memory.

Decoupled synchronization and multiple timescales. DES-LOC assigns independent synchronization periods to parameters and optimizer states in data-parallel training and shows that states with longer half-lives can be synchronized less often (Iacob et al., 2026). Its workers optimize a shared objective with full participation, so their optimizer states differ mainly through gradient noise. In FL, heterogeneous objectives and intermittent participation make optimizer states client-specific between rounds, so the useful synchronization timescale depends on the federated setting as well as on the optimizer's decay rates. Nested Learning views learning systems as components updated at different frequencies (Behrouz et al., 2025), and Federated Nested Learning studies collaborative learned memories for test-time adaptation (Chen et al., 2026). Our focus is narrower: we keep the model on an every-round clock and study the cross-client synchronization timescale of conventional optimizer memories.

Communication-efficient and adaptive synchronization. Communication-efficient FL reduces cost through periodic averaging with quantization (Reisizadeh et al., 2020), gradient quantization (Alistarh et al., 2017), and sparsification (Stich et al., 2018). FedLAMA adapts aggregation intervals across layers based on layer-wise model discrepancy (Lee et al., 2023), and recent work reduces the communication cost of adaptive optimization in FL (Deng et al., 2025). These methods adapt how often or how precisely a predefined object is communicated. We instead measure whether an aggregated optimizer memory is still useful when delivered, and use that measurement to set its synchronization interval.