PARALLAX: Hierarchical Sparse Pre-training Across Fragmented Compute

Jon Durbin

June 2026

Abstract

PARALLAX is a training decomposition for sparse Mixture-of-Experts models on heterogeneous GPUs that are not colocated. Each composer owns a disjoint shard of routed experts and keeps detached low-rank surrogates for the rest. The router still selects across the global expert namespace, but the MoE layer runs locally: owned selections call real experts, non-owned selections call surrogates, and no token-level expert all-to-all sits on the step path. Owners refresh surrogate state in the background. Router, latent-interface, and backbone parameters synchronize through tiered RDA-DiLoCo-style cadences. A worker-offload variant moves routed-expert optimizer state and expert update work to remote GPUs using compressed activation sketches and Taylor-proxy expert objectives.

The report gives point estimates from completed 20B runs. It does not report replicated equivalence tests or measured wall-clock speedups. A 4-composer run on non-colocated H100 GPUs reaches median validation loss 3.1896 at the 30k baseline-equivalent comparison; the interpolated 4×B300 end-to-end baseline is 3.1990 at the same token count. An 8-composer L40S/NVIDIA RTX 6000 Ada run with remote NVIDIA RTX 4090 expert workers reaches 3.2521 at 50k baseline-equivalent steps and a best exported median of 3.2104 after further training. For the same 20B architecture, a rank-64 surrogate uses 48× fewer expert hidden dimensions than a real routed expert. At $C = 8$, the analytic model gives 7.6× lower routed-expert training compute and 2.16× lower total active compute for direct ownership; with expert offload, it gives 20.9× lower composer-side routed-expert compute and 2.43× lower total active compute.

1 Introduction

Large-model training is increasingly limited by the availability of tightly coupled accelerator clusters. Sparse Mixture-of-Experts (MoE) models appear modular because only a subset of experts is active for each token, but existing expert-parallel systems still assume a fast collective fabric: token activations are dispatched to remote expert owners at each MoE layer and returned before the step can continue. The model is sparse; the training system is still built as if every participant sits behind a datacenter interconnect.

This is the wrong shape for fragmented compute. Useful GPUs may be single machines, small local clusters, or temporary worker fleets. Their memory limits and bandwidth profiles vary; so do reliability and operating cost. A training method for that environment should not merely tolerate smaller participants; adding participants should reduce the minimum burden carried by each one.

This paper asks a systems question: can a sparse MoE train against every expert id without executing remote experts on the critical path? The challenge is not simply to shard parameters. A composer must still make routing decisions over all experts, preserve the router/expert contract, update experts from useful training signal, and synchronize the shared parts of the model often enough that independently trained composers do not drift into incompatible models.

PARALLAX changes the unit of responsibility. Each composer owns only $1/C$ of routed experts, stores those experts at full fidelity, and substitutes detached low-rank surrogates for the other $(1 - 1/C)$ experts. Tokens still route over all experts. The local step no longer waits for remote expert execution. Non-owned experts provide forward context through surrogates; their real updates come from their owners or from expert workers. The non-expert model state synchronizes on tiered cadences.

The immediate payoff is narrow and mechanical. Non-owned routed experts have no composer-side backward pass. Expert all-to-all leaves the synchronous step. Routed-expert optimizer state scales with ownership and can leave the composer entirely in the offload variant. These gains come from the decomposition itself.

This is hierarchical sparse pre-training: the router sees one global sparse model, but training work is stratified into owned experts, surrogate recall, tiered shared-state synchronization, and optional worker-side expert updates. In the analytic model, increasing composer count lowers per-composer expert compute and state while increasing the communication burden of conventional expert parallelism. The resulting advantage can grow faster than any single local saving, though total active-compute gains remain bounded by the always-active backbone and are not reported as measured wall-clock speedups.

Table 1: Status of the main claims in this report.

Claim	Evidence	Status
C=4 20B PARALLAX tracks the end-to-end baseline through the 30k-equivalent comparison.	Completed 20B run, single-run baseline, token-axis interpolation.	Empirical point estimate.
C=8 worker-offload trains a 20B model with cheaper composer hardware and remote workers.	Completed 20B run with L40S/NVIDIA RTX 6000 Ada composers and NVIDIA RTX 4090 workers.	Empirical point estimate in a noisier setting.
Expert all-to-all is removed from the local step path.	Owned experts plus detached local surrogates for non-owned experts.	Direct property of the algorithm.
Routed-expert compute and composer state fall with C.	FLOP and Adam-state accounting for the measured architecture.	Analytic model; wall-clock speedup is unmeasured.
176B architecture can launch under the same abstraction.	135-step smoke test on four B300 nodes.	Launch-path feasibility only.

2 Training Problem

PARALLAX targets decentralized sparse-MoE training on heterogeneous accelerators connected by commodity networks. The training method must preserve the modeling behavior of a global MoE while avoiding the communication pattern that ordinary expert parallelism requires.

This yields four design requirements.

1. **Local completeness**: every composer must be able to execute a full forward pass over the global expert namespace without remote expert dispatch.
2. **Expert-state locality**: no composer should need optimizer state for all routed experts.
3. **Asynchronous expert work**: expert updates must be decomposable into tasks suitable for cheaper worker GPUs.
4. **Tiered synchronization**: router and latent parameters need tighter synchronization than backbone blocks, surrogate state, or owned experts.

PARALLAX addresses these requirements through owned experts, low-rank surrogate recall, tiered RDA-DiLoCo, and worker-offloaded expert updates. Inference-efficient architecture choices are discussed only as future work; they are not required for the core training result.

3 Related Work

3.1 Sparse MoE Training

Sparse Mixture-of-Experts models increase total parameter capacity while keeping active compute per token sublinear in total parameters [Shazeer et al., 2017, Fedus et al., 2022, Lepikhin et al., 2021, Du et al., 2022, Jiang et al., 2024, DeepSeek-AI, 2024, Dai et al., 2024, Muennighoff et al., 2024, Krajewski et al., 2024, Qwen Team, 2025]. Modern sparse MoE systems route each token to a subset of expert MLPs, typically using top-k routing and auxiliary, bias-based, hash, or expert-choice load-balancing mechanisms [Fedus et al., 2022, Zoph et al., 2022, Zhou et al., 2022, Clark et al., 2022, Lewis et al., 2021]. The practical challenge is that expert sparsity helps arithmetic but creates a systems problem: tokens assigned to remote experts must cross the fabric and return before the layer can finish. This has shaped MoE infrastructure toward tightly coupled expert-parallel clusters with high-bandwidth all-to-all fabrics [Rajbhandari et al., 2022, Hwang et al., 2023, Gale et al., 2023, He et al., 2021, Xue et al., 2024].

PARALLAX preserves the modeling advantage of a global expert namespace while changing the physical execution rule. Rather than dispatching token activations to wherever an expert lives, each composer executes owned experts exactly and non-owned experts through local surrogates. The router still sees the full expert set. The model still benefits from sparse capacity. The synchronous systems requirement changes from remote expert execution to local surrogate recall plus asynchronous refresh.

3.2 Distributed and Decentralized Training

Data-parallel training and modern sharded training systems such as ZeRO, FSDP, tensor parallelism, and model parallelism are highly effective when the cluster is colocated and bandwidth is predictable [Rajbhandari et al., 2020,

Zhao et al., 2023]. They are not designed for a setting where useful accelerators are distributed across the internet, have different memory sizes, and cannot participate in a tight all-reduce on every step. Local-SGD, federated optimization, and DiLoCo-style methods reduce synchronization frequency by allowing workers to take multiple local steps before exchanging pseudo-gradients or model deltas [Stich, 2019, McMahan et al., 2017, Li et al., 2020, Karimireddy et al., 2020, Douillard et al., 2023, Jaghouar et al., 2024, Ryabinin and Gusev, 2020, Diskin et al., 2021, Prime Intellect, 2024, Jaghouar et al., 2025, Senghaas et al., 2025, Nous Research, 2025, Peng, 2024, Lidin et al., 2026]. Streaming and decoupled variants overlap communication with computation and compress or tier the exchanged state [Douillard et al., 2025a,b].

PARALLAX is complementary to that line of work. It uses tiered RDA-DiLoCo for non-expert parameters, but it does not treat the MoE as a monolithic parameter vector. Router state, latent projections, backbone blocks, surrogate banks, and routed experts have different synchronization tolerances and different physical roles. Expert execution is localized, surrogate state is refreshed by owners, and only non-expert parameters require model-level averaging.

3.3 Gradient Compression and Low-Rank Updates

Distributed optimizers reduce communication through top-k sparsification, error feedback, low-rank compression, sign methods, quantization schemes, and rotation-based transforms [Lin et al., 2018, Stich et al., 2018, Karimireddy et al., 2019, Vogels et al., 2019, Alistarh et al., 2017, Bernstein et al., 2018, Tang et al., 2021, Richtárik et al., 2021, Ashkboos et al., 2024b, Wang et al., 2024]. In ordinary full-model synchronization, compression is applied after the payload has already been defined as a large model delta or gradient. PARALLAX first changes the payload itself: the exchanged object is an expert-level update, a surrogate delta, or an activation sketch rather than a full-model gradient. Low-rank surrogate updates are especially attractive because the surrogate itself is low rank; a rank-q delta to two surrogate matrices is naturally small and can be sent as a background control message. Low-rank adaptation and gradient projection methods provide related tools, but PARALLAX uses the low-rank object as a callable surrogate for a non-owned expert rather than as a parameter-efficient fine-tuning adapter [Hu et al., 2022, Zhao et al., 2024].

3.4 Asynchronous and Stale Updates

Asynchronous SGD, stale-synchronous methods, parameter servers, and sparse Hogwild updates all exploit the fact that not every parameter update must be globally synchronized at the same instant [Recht et al., 2011, Ho et al., 2013, Li et al., 2014, Zheng et al., 2017, Mishchenko et al., 2022]. MoE experts are a particularly favorable target for this idea because each routed expert sees only a subset of tokens and has a natural local objective. PARALLAX uses this structure at the expert boundary. Worker updates are bounded, audited by local proxy objectives, and applied asynchronously; meanwhile router and backbone synchronization remain tiered because they affect all tokens.

3.5 Synthetic Gradients and Local Objectives

Synthetic-gradient methods decouple modules by learning gradient predictors [Jaderberg et al., 2017]. PARALLAX does not learn a standalone gradient predictor. Workers optimize against real activation sketches captured during ordinary composer training. The Taylor proxy is a local first-order objective induced by the actual upstream expert-output gradient. This is a narrower and more auditable form of decoupling: the worker receives neither the task nor a data shard, and it never trains a full model; it improves an expert against a compact local approximation of the loss.

4 Method

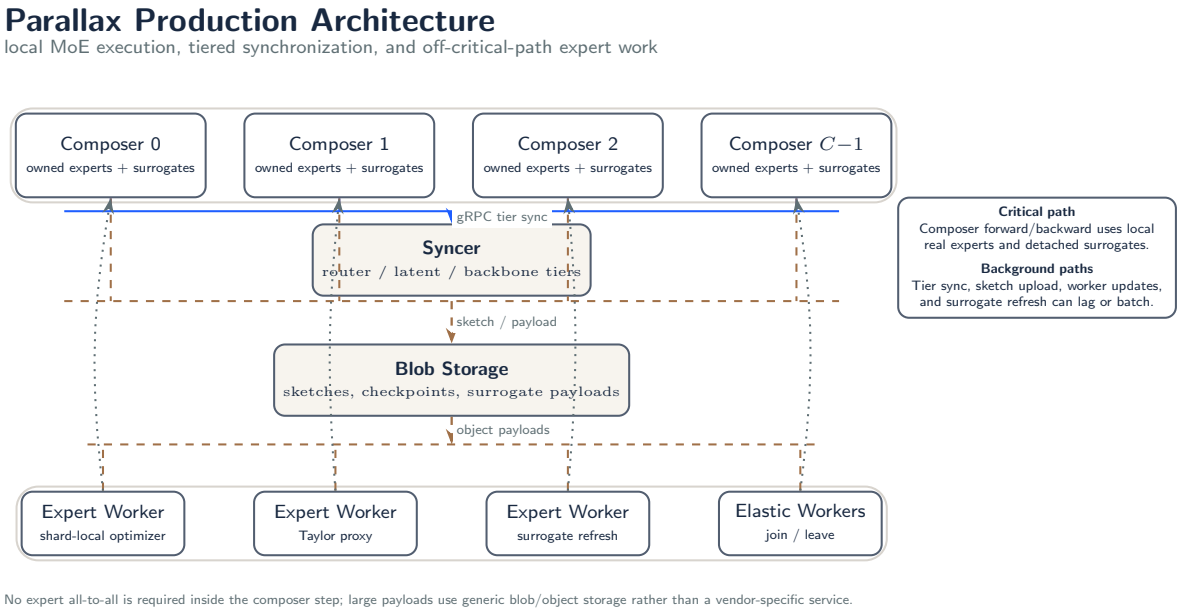

Figure 1: PARALLAX production architecture. Composers execute locally complete MoE steps with owned experts and surrogate recall; a syncer coordinates tiered non-expert state over gRPC; blob storage carries large artifacts such as sketches/checkpoints/surrogate payloads; expert workers process sketch-derived Taylor-proxy updates off the critical path.

4.1 Expert Ownership

Let model parameters be partitioned into backbone parameters θ_B, router parameters θ_R, latent projection parameters θ_L, and routed expert parameters θ_E. For each MoE layer with N routed experts and C composers, composer c owns

$$\mathcal{O}_c = \{e \in \{0, \dots, N-1\} : e \bmod C = c\}.$$

The ownership map can be round-robin, hashed, or load-aware; the tested large runs use hard disjoint ownership. Hard ownership keeps the experiment strict: there is no soft blend between real and surrogate experts.

For an MoE layer, the ordinary routed output is

$$y(x) = \sum_{e \in \mathrm{TopK}(x)} w_e(x) E_e(x),$$

where E_e is the routed expert and w_e is the routing weight. Composer c instead executes

$$y_c(x) = \sum_{e \in \mathrm{TopK}(x)} w_e(x) \begin{cases} E_e(x), & e \in \mathcal{O}_c, \\ \mathrm{stopgrad}(\hat{E}_e(x)), & e \notin \mathcal{O}_c. \end{cases}$$

The stop-gradient barrier prevents surrogate error from becoming an uncontrolled training signal through approximate non-owned experts. Owned experts receive ordinary updates when trained on-composer. Non-owned experts contribute forward signal only, with their true updates supplied by their owners or by expert workers.

4.2 Low-Rank Surrogates

The 20B model uses LatentMoE experts [Elango et al., 2026]. Inputs are projected from model width into an expert latent space of dimension $d_e = 768$, routed experts operate in that latent space, and outputs are projected back to model width. A relu^2 routed expert has two matrices and hidden width $h = 3072$:

$$E_e(x) = \mathrm{ReLU}(xW_{1,e})^2 W_{2,e},$$
$$W_{1,e} \in \mathbb{R}^{d_e \times h}, \qquad W_{2,e} \in \mathbb{R}^{h \times d_e}.$$

The surrogate has the same input/output interface but hidden rank r:

$$\hat{E}_e(x) = \mathrm{ReLU}(xA_e)^2 B_e, \qquad A_e \in \mathbb{R}^{d_e \times r},\ B_e \in \mathbb{R}^{r \times d_e}.$$

For $r = 64$, the surrogate has $2d_e r$ parameters versus $2d_e h$ for the real expert, so the surrogate forward is $h/r = 48$ times smaller before considering backward-pass removal.

Owners recalibrate surrogates for their experts from current expert weights and recent activation statistics. A full surrogate replacement is possible but unnecessary. In the intended steady state, an owner sends a low-rank update to the surrogate parameters. If the update rank is q, each two-matrix surrogate update costs approximately

$$2q(d_e + r)$$

scalars per expert, before compression. This makes surrogate update traffic a background control channel rather than a token-by-token dispatch fabric.

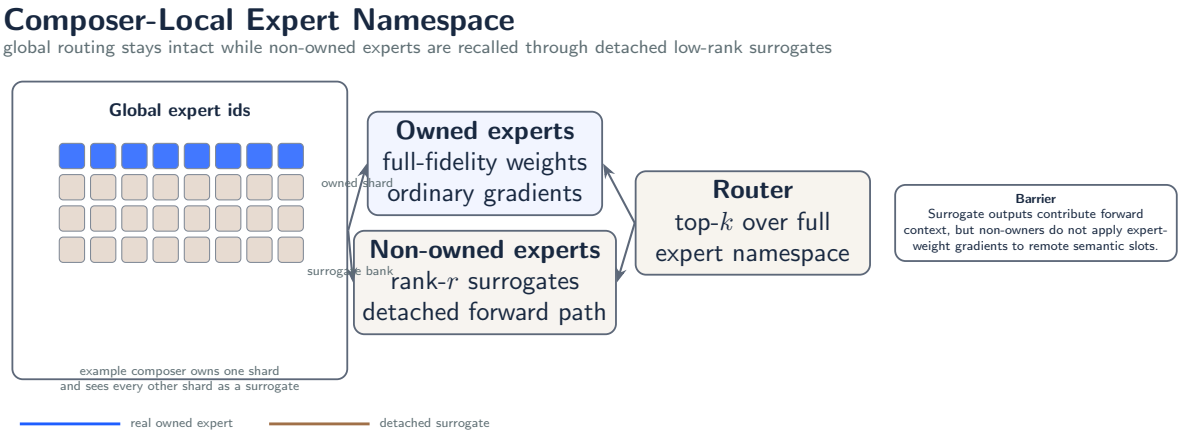

Figure 2: Composer-local view of routed experts. Each composer owns a shard at full fidelity and stores compact surrogates for the remaining expert namespace, so routing remains global while execution stays local.

4.3 Avoiding Expert Semantic Collision

A sparse MoE has a hidden distributed-systems problem: the expert id is both a parameter shard and a semantic slot learned jointly by the router and the expert. In ordinary decentralized training, different islands can route different local data distributions into the same nominal expert id, update that expert under different local semantics, and later average the resulting weights. The average may be numerically well defined while being semantically destructive: two specialists with incompatible routed-token histories are merged into a single expert.

PARALLAX avoids this failure mode by separating global expert visibility from expert authority. Every composer routes across all expert ids, so the router can learn a global allocation problem. But only an owner, or a worker acting from owner-approved sketches and versions, applies authoritative updates to the real expert. Non-owners see the expert through a surrogate forward path and do not backpropagate a private local expert update into that remote semantic slot. Frequent router and latent-interface synchronization then keeps composers aligned about what the expert ids mean.

The goal is not for a composer to route only to its own experts. That would collapse the global MoE into private local MoEs. All composers should route globally while real expert updates stay tied to a shard owner and a version. Surrogates provide the missing local completeness: a composer can evaluate the consequence of selecting any expert without waiting for remote execution or mutating that expert from a private local loss.

4.4 Tiered RDA-DiLoCo Synchronization

The non-expert parameters are synchronized at different cadences. Router and latent projection parameters are sensitive to composer drift because they define which experts are selected and what interface the experts see. They are synchronized frequently. The rest of the backbone is synchronized less frequently and can be fragmented so transfer overlaps local training.

The tested implementation uses a syncer node as a coordination point. Composers periodically publish parameter deltas or merged states to the syncer, and the syncer returns a consolidated state according to the tier being synchronized. This is a decoupled DiLoCo-style protocol: it is not synchronous data parallelism, and it does not require all composers to participate in every local step. The 20B method reported here is routed-expert surrogate substitution plus tiered RDA-DiLoCo synchronization.

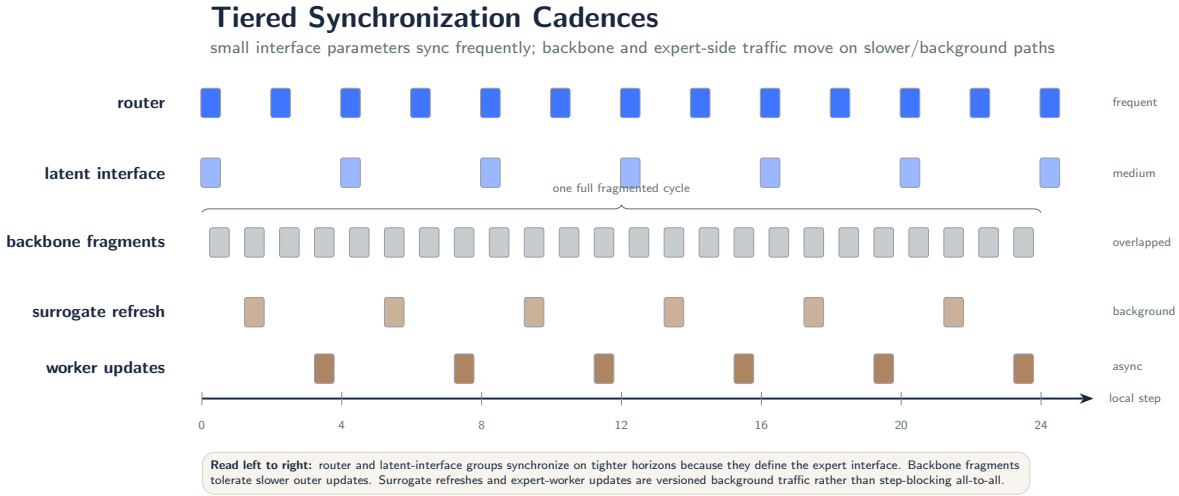

Figure 3: Tiered synchronization timeline. Router and latent-interface parameters synchronize more frequently than backbone fragments, while surrogate and expert-worker traffic remain background channels.

4.5 Worker-Offloaded Expert Updates

The offloaded variant moves routed-expert optimizer state and much of the expert update work away from composers. During normal forward/backward execution, a composer records activation sketches: expert inputs in latent space, selected expert ids, routing weights, upstream output gradients, and enough metadata to reconstruct a local expert objective. Sketches are deduplicated and compressed; the current implementation supports aggressive 4-bit/HQ4-style compression for sketch payloads, drawing on recent low-bit and rotation-based quantization work [Ashkboos et al., 2024a,b, Zandieh et al., 2025, Frantar et al., 2023, Lin et al., 2024, Chee et al., 2024].

Workers consume sketches for one or a small number of experts. They optimize a Taylor-proxy objective of the form

$$\Delta\mathcal{L}_e \approx \langle g_{y,e}, E_e(x;\theta_e+\Delta\theta_e) - E_e(x;\theta_e)\rangle + \frac{\lambda}{2}\|\Delta\theta_e\|_2^2,$$

where $g_{y,e}$ is the captured upstream gradient for the expert output. The worker can operate with one expert, a sketch batch, and local optimizer buffers. It returns an expert update and, when appropriate, a refreshed surrogate update. This makes a heterogeneous fleet useful: a worker GPU can be much smaller than the composer GPU because it only needs a shard of the routed-expert problem.

4.6 LatentMoE and Ternary Expert Design

Parallax does not mathematically require LatentMoE or ternary experts, but both are important design choices in the tested system. LatentMoE makes the expert interface compact: sketches, surrogate matrices, and expert all-to-all accounting scale with d_e rather than full model width [Elango et al., 2026]. It also makes surrogate rank meaningful; a rank-64 surrogate in a 768-dimensional latent space is a different proposition from approximating full-width expert functions.

Ternary routed experts compound the training-system savings by reducing expert storage and lowering memory pressure for owned shards [Li et al., 2016, Zhu et al., 2017, Wang et al., 2023, Ma et al., 2024]. The cost model reports conservative Adam-state accounting because optimizer moments can remain higher precision even when the expert forward representation is low bit [Kingma and Ba, 2015, Loshchilov and Hutter, 2019].

4.7 Composer Step Protocol

The local composer step has three distinct update paths: ordinary gradients for non-expert parameters, owned-expert updates or owned-expert forward capture, and background metadata capture for surrogates/workers. The following protocol is the conceptual step executed by composer c.

Protocol 1: Composer local step

1. Receive a token batch and compute embeddings/backbone activations through the next MoE layer.
2. Score the global expert list and select top-k experts per token.
3. For each selected expert e: execute E_e when $e \in \mathcal{O}_c$; otherwise execute detached $\hat{E}_e$.
4. Accumulate backbone, router, latent-projection, and owned-expert gradients according to the selected mode.
5. Capture activation sketches for routed experts targeted by worker offload: latent inputs, selected expert ids, routing weights, output gradients, step metadata, and version identifiers.
6. Apply the local optimizer step for trainable on-composer parameters.
7. Enqueue tier-specific synchronization events when their cadence triggers.
8. Enqueue surrogate refresh or worker dispatch events independently of the critical path.

Steps 1–6 can complete without waiting on any remote expert. Background queues may lag temporarily; the training step does not depend on token activations crossing an expert-parallel fabric.

4.8 Surrogate Refresh Protocol

Surrogate refresh is owner-driven. Each expert has a single authoritative owner in the hard-barrier implementation. The owner periodically fits or updates the surrogate representation for that expert and publishes the update to non-owners. Non-owners install those updates as refreshed expert state; their local losses do not directly train non-owned surrogates.

Protocol 2: Owner surrogate refresh

1. Select owned expert e whose surrogate version is due for refresh.
2. Build a calibration batch from recent latent expert inputs and the current real expert output.
3. Fit a rank-r surrogate or compute a rank-q delta to the existing surrogate.
4. Attach expert id, layer id, owner id, model-step version, and checksum metadata.
5. Broadcast the compact surrogate update to all non-owner composers.
6. Non-owners atomically swap or apply the update between local training steps.

This protocol gives every composer a locally executable approximation of every non-owned expert while keeping the true expert state authoritative. It also prevents multiple composers from independently drifting the same surrogate in incompatible directions.

4.9 Tiered Synchronization Protocol

Parameter groups are synchronized by sensitivity. Let H_R, H_L, and H_B denote router, latent-projection, and backbone horizons. Typically $H_R \leq H_L \leq H_B$. The syncer holds the outer optimizer state for the synchronized group and returns a consolidated state or delta.

Protocol 3: Tiered RDA-DiLoCo synchronization

1. At each local step, composer c updates its local parameters normally.
2. On router-sync steps ($t \bmod H_R = 0$), publish router delta $\Delta\theta_R^{(c)}$.
3. On latent-sync steps ($t \bmod H_L = 0$), publish latent-projection delta $\Delta\theta_L^{(c)}$.
4. On backbone-sync steps ($t \bmod H_B = 0$), publish the next backbone fragment delta $\Delta\theta_{B,j}^{(c)}$.
5. The syncer merges received deltas with an outer optimizer and exposes the updated tier state.
6. Composers pull the latest available state for each tier and apply it at safe local boundaries.

Fragmenting the backbone tier prevents a large backbone sync from becoming a bursty network event. Fragmentation turns the backbone into a sequence of smaller transfers that can be overlapped with local training. Router and latent tiers stay frequent because stale routing decisions create a sharper cross-composer mismatch than stale deep backbone weights.

4.10 Activation Sketch Format

An activation sketch gives a worker the local information needed to improve one expert without holding the full model. For expert e in layer ℓ, a sketch contains

$$S_{e,\ell} = \{X_{e,\ell},\ G_{e,\ell},\ w_{e,\ell},\ m_{e,\ell}\},$$

where $X_{e,\ell} \in \mathbb{R}^{n_e \times d_e}$ are latent expert inputs, $G_{e,\ell} \in \mathbb{R}^{n_e \times d_e}$ are upstream output gradients, $w_{e,\ell}$ are routing weights, and $m_{e,\ell}$ records the step, expert version, and normalization metadata. The sketch does not contain raw text tokens or full model activations outside the expert interface.

Sketch compression is especially favorable in LatentMoE because d_e is much smaller than full model width. The payload is also naturally deduplicated: repeated expert/layer/version combinations can reuse weights, calibration metadata, or cached sketches. In the offloaded setting, sketch traffic replaces composer-side routed-expert backward passes and routed-expert Adam state.

4.11 Worker Update Protocol

Protocol 4: Worker expert update

1. Pull an expert assignment, current expert weights, and one or more fresh sketches.
2. Decompress sketches and reconstruct the local Taylor-proxy objective.
3. Optimize the expert update with a proximal trust term and bounded inner-loop budget.
4. Compress the resulting expert delta or updated expert state.
5. Report proxy-loss improvement, update norms, version metadata, and optional surrogate refresh data.
6. Composer or owner applies the update if version and acceptance checks pass.

Workers do not synchronize with one another. Each worker only needs enough current expert state and sketch context to produce a useful local update. Heterogeneity follows from that locality: a worker may be assigned fewer experts, smaller batches, or less frequent work according to its memory and bandwidth.

4.12 Update Acceptance and Versioning

Asynchrony requires version discipline. Every sketch and expert update is tagged with a model-step version, expert version, owner id, and layer id. Updates are accepted only when their version lag is within the configured tolerance and their local proxy diagnostics are well behaved. The simplest acceptance gate is monotonic proxy improvement under a proximal penalty:

$$\mathcal{P}_e(\theta_e + \Delta\theta_e; S_{e,\ell}) < \mathcal{P}_e(\theta_e; S_{e,\ell}),$$

where $\mathcal{P}_e$ is the worker proxy objective. Stronger gates can include norm caps, spectral caps, redundant worker comparison, or sampled full-model checks. The present paper treats these as systems safeguards; the core algorithmic result is the expert ownership and surrogate decomposition.

5 Cost Model

For a relu^2 expert with two linear layers, a forward call costs

$$F_{\text{real}} = 4d_e h$$

FLOPs per token-expert assignment when multiply-add is counted as two FLOPs. Expert training with backward and weight gradients costs approximately $3F_{\text{real}}$. A detached surrogate forward costs

$$F_{\text{surr}} = 4d_e r$$

and has no backward pass on the composer.

Composer-side routed-expert compute relative to ordinary end-to-end expert training is

$$\rho_{\text{direct}}(C) = \frac{1}{C} + \left(1 - \frac{1}{C}\right)\frac{r}{3h}$$

when composers directly train owned experts, and

$$\rho_{\text{offload}}(C) = \frac{1}{3C} + \left(1 - \frac{1}{C}\right)\frac{r}{3h}$$

when owned experts are forward-only on composers and expert optimization is offloaded.

The offload expression is a composer-side accounting model. It counts the owned-expert forward needed to produce the local MoE output, but not composer-side expert weight gradients or Adam updates. In this variant, the authoritative routed-expert optimization work is performed on workers from captured sketches. Gradients for router, latent interface, and backbone parameters are still computed through the local forward graph; the accounting reduction is specifically for routed-expert training compute and state on the composer.

For the measured 20B configuration, $r/h = 64/3072 = 1/48$. At $C = 4$, direct composer training uses 25.5% of ordinary routed-expert training compute; this corresponds to a 3.9× routed-expert reduction and 1.85× total active-compute reduction. At $C = 8$, direct ownership uses 13.1% of ordinary routed-expert training compute, giving 7.6× on routed experts and 2.16× end-to-end active arithmetic. With worker offload at $C = 8$, composer-side routed-expert training compute falls to 4.77%, or 20.9× on that component and 2.43× for total active compute. The smaller total-compute numbers are the right numbers for end-to-end arithmetic; the larger numbers describe only the routed-expert portion or the composer-side routed-expert portion.

The same analysis bounds memory. With bf16 parameters and gradients plus fp32 Adam master copies and two moment buffers, ordinary Adam training state is approximately 16 bytes per trainable parameter. Direct ownership reduces routed-expert optimizer state by C. Worker offload removes routed-expert optimizer state from composers entirely. Composer expert weight storage is

$$M_E(C) \propto \frac{1}{C} + \left(1 - \frac{1}{C}\right)\frac{r}{h},$$

because non-owned experts are stored as surrogates.

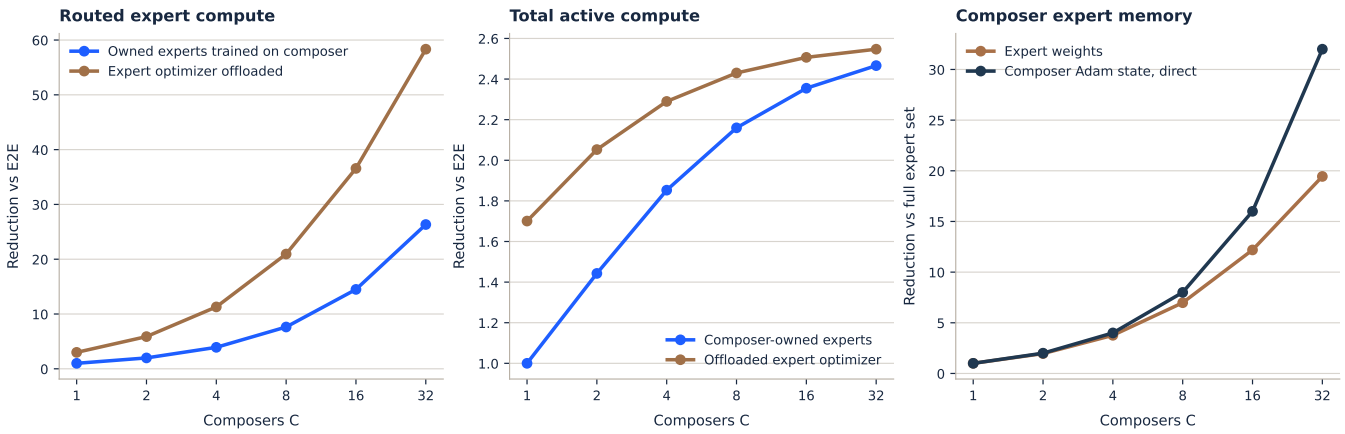

Figure 4: 20B theoretical savings from surrogate substitution and expert offload. End-to-end active compute savings are bounded by the non-expert backbone share, but routed-expert compute and memory scale strongly with composer count.

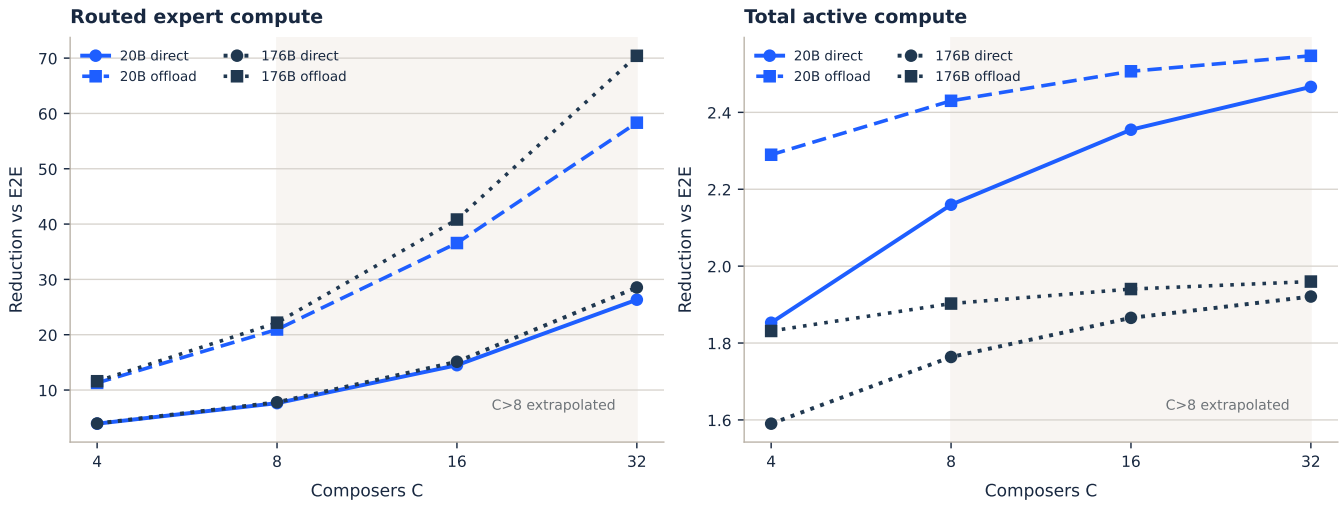

Figure 5: Compute reductions for the measured 20B architecture and projected 176B design. Design rows are extrapolations from the architecture plan and are not claimed as completed convergence results.

Table 2: Cost-model assumptions. Measured rows are tied to the 20B experiments; design rows are extrapolations used only for scaling analysis.

Model	Status	Total	Active	d_e	h	r
20B	measured	20.0B	1.95B	768	3072	64
176B	feasibility/design	176.3B	8.65B	1024	4096	48

Table 3: Theoretical reductions at the tested and proposed composer counts. Expert compute excludes backbone; total compute includes the active backbone share.

Model, C	Surr. fwd	Expert direct	Expert offload	Total direct	Total offload
20B, 4	48.00×	3.92×	11.29×	1.85×	2.29×
20B, 8	48.00×	7.63×	20.95×	2.16×	2.43×
20B, 16	48.00×	14.49×	36.57×	2.35×	2.51×
176B, 4	85.33×	3.95×	11.59×	1.59×	1.83×
176B, 8	85.33×	7.79×	22.18×	1.76×	1.90×
176B, 16	85.33×	15.11×	40.82×	1.87×	1.94×

Table 4: Per-composer expert memory and critical-path communication accounting under a common global-batch assumption for scaling comparison. Training state assumes bf16 parameters/gradients plus fp32 Adam master, first moment, and second moment; experiment-specific 20B communication values are stated in the text.

Model, C	Expert weights	Direct expert state	Expert A2A @ B=1024	Rank-8 surr. sync
20B, 4	10.3 GB	77.2 GB	618 GB/step	81.8 MB
20B, 8	5.5 GB	38.6 GB	722 GB/step	95.4 MB
20B, 16	3.2 GB	19.3 GB	773 GB/step	102.2 MB
176B, 4	89.0 GB	687.6 GB	1649 GB/step	526.9 MB
176B, 8	46.5 GB	343.8 GB	1924 GB/step	614.7 MB
176B, 16	25.3 GB	171.9 GB	2062 GB/step	658.6 MB

5.1 Communication

In ordinary expert parallelism, remote expert inputs and outputs must be exchanged for every MoE layer. Under balanced routing and the tested ownership layouts, the expected synchronous expert all-to-all traffic per optimizer step

is at least

$$2 \cdot L_{\text{moe}} \cdot B \cdot S \cdot k \cdot \left(1 - \frac{1}{C}\right) \cdot d_e \cdot \text{bytes}.$$

The factor of two counts inputs and outputs. This deliberately narrow accounting identity excludes routing metadata, capacity padding, load-imbalance overhead, collective protocol overhead, and buffering; it should not be read as a throughput measurement.

Applying the formula to the two 20B runs gives configuration-specific activation byte counts: 309 GB/step for the H100 setting ($C = 4$, global batch 512, sequence length 1024) and 722 GB/step for the L40S/NVIDIA RTX 6000 Ada worker-offload setting ($C = 8$, global batch 1024). The values differ because the runs use different global token rates and composer counts; they are separate accounting points rather than an uncertainty interval. PARALLAX removes this synchronous expert all-to-all from the critical path. Surrogate refresh and worker traffic move to asynchronous background channels, with compact payloads and no per-step token dispatch.

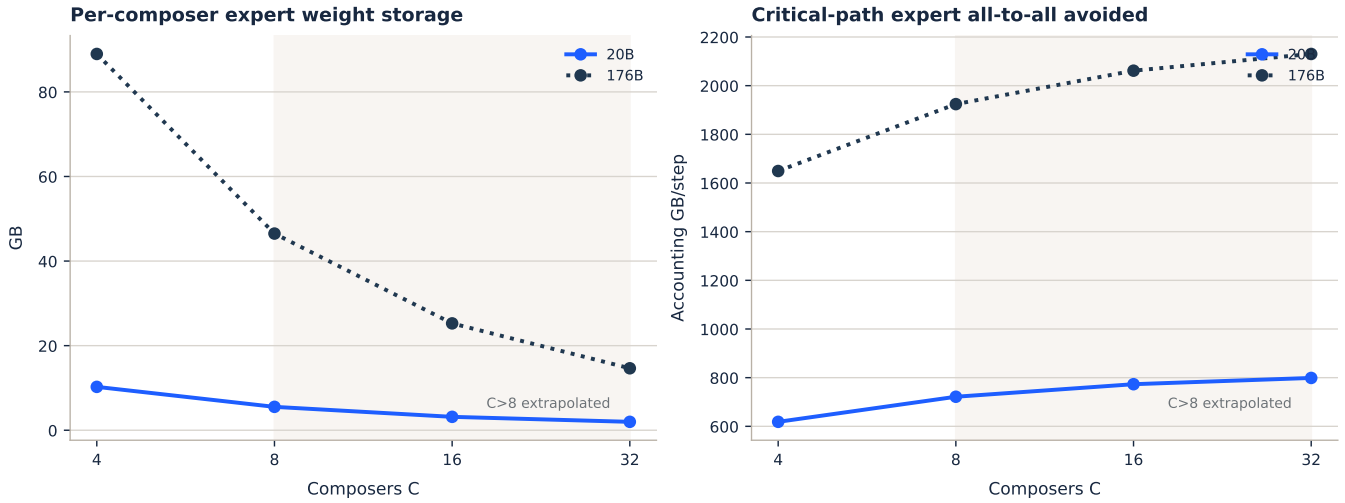

Figure 6: Per-composer expert storage and synchronous expert all-to-all accounting under the scaling model. The communication plot uses a common batch assumption for comparison across model designs; experiment-specific 20B byte counts are stated in the text.

5.2 Worker-Side Memory Accounting

Worker memory is dominated by one or a small number of expert weights, optimizer state for those experts, and sketch batches. For a two-matrix relu^2 expert with latent dimension d_e and hidden width h, the expert has $2d_eh$ parameters. A worker training that expert with Adam-style state requires approximately

$$16 \cdot 2d_eh$$

bytes for bf16 parameters/gradients and fp32 Adam master/moment state, before framework overhead. For the 20B expert shape ($d_e = 768$, $h = 3072$), this is 4.72M parameters and about 75.5 MB of training state per expert. Even with activation buffers, sketch batches, and temporary optimizer memory, the worker problem is far smaller than holding the full 20B model or its optimizer state.

A composer hosts the backbone, router, latent projections, owned experts, and non-owned surrogates. A worker only needs the assigned expert state and sketches. Scaling total model capacity by adding more experts increases the number of possible worker assignments; it does not force each worker to hold the full model.

5.3 Critical-Path Versus Background Traffic

Communication savings should be separated into critical-path and background traffic. Expert all-to-all in standard expert parallelism is critical-path traffic: the next layer cannot proceed until remote expert outputs arrive. PARALLAX removes this path entirely. Surrogate refresh, worker sketch dispatch, worker update return, and tiered model synchronization are background or cadence-bound traffic. They can be delayed, batched, compressed, or skipped without blocking the immediate local forward pass.

Raw byte counts miss the scheduling problem. A 100 MB background transfer every few steps can be delayed, retried, or batched. Hundreds of GB of token dispatch inside a layer cannot; the next layer is waiting. Background traffic can use commodity internet links and object-store-like fanout. Critical-path token dispatch wants low-latency, high-bandwidth, low-jitter interconnect.

5.4 Scaling With Composer Count

Composer count C creates three separate scaling effects. The first is owned expert fraction: each composer trains or stores only $1/C$ of full experts. The second is surrogate fraction: each composer still stores approximations for $(1-1/C)$ of experts, but the approximation ratio is r/h. The third is remote-token fraction in an ordinary expert-parallel baseline: all-to-all traffic grows with $(1-1/C)$ because more selected experts are remote as the model is distributed across more shards.

Under balanced routing, larger C reduces PARALLAX composer memory and routed-expert compute while increasing the expected remote-token traffic of ordinary expert parallelism. The cost model improves as the expert pool is sharded more aggressively. The optimization limit is separate: stale surrogates, router drift, owner-specific data skew, and worker update latency can dominate before the algebraic limit is reached.

In that limited accounting sense, the advantage is superlinear in composer count: PARALLAX reduces the local routed-expert burden as ownership is split, while the ordinary expert-parallel comparator sends a larger fraction of selected expert traffic off shard. The term applies to the combined expert-compute, expert-state, and critical-path communication model. Measured wall-clock speedup is a separate question.

5.5 Ternary Storage Versus Training State

Ternary experts reduce stored expert weights, but training-state accounting must be stated carefully. If an expert is trained with Adam, the optimizer may still maintain higher-precision master copies and optimizer moments even when the forward representation is ternary or low-bit. Therefore the memory tables report ordinary training-state assumptions for a conservative comparison. The inference and checkpoint storage advantages of ternary experts are additional benefits, especially when expert weights can be packed and served through FP4 or ternary-aware kernels.

In the offloaded variant, the distinction becomes sharper. Composer-side routed-expert Adam state is zero regardless of whether workers use full-precision, 8-bit, or low-bit optimizer state internally. Worker optimizer precision changes worker memory and bandwidth while leaving composer feasibility unchanged.

6 Experimental Protocol

All quantitative results use a completed-run corpus with exact-content deduplication before tabulation. The manuscript reports experiment settings and metrics; internal logging tools and storage locations are not part of the report.

The main 20B comparison has three arms. The baseline is an end-to-end 20B run across a four-B300 setup using the matched architecture family. The C=4 PARALLAX run uses four single-H100 composers distributed across a wide-area network, with direct owned-expert training and no worker offload. The C=8 PARALLAX run uses L40S/NVIDIA RTX 6000 Ada composers plus remote NVIDIA RTX 4090 expert workers, with routed-expert update work offloaded through sketches.

Table 5: 20B training recipe used for the PARALLAX arms. The baseline uses the matched 20B family in an end-to-end four-B300 run; Table 9 reports token-matched comparisons.

Item	Value
Dataset and tokenizer	FineWeb 100B tokenized with the Llama 3 tokenizer; vocabulary 128,256.
Architecture preset	20B LatentMoE with ternary routed experts.
Depth and block mix	32 layers: 16 attention blocks and 16 MoE blocks.
Model width and sequence length	$d_{\text{model}} = 2048$; training sequence length 1024; configured maximum length 2048.
Expert layout	256 routed experts per MoE layer, top-$k = 16$, no shared experts.
Latent expert shape	latent dimension 768; relu^2 expert $768 \to 3072 \to 768$.
Surrogate shape	rank 64 for non-owned routed experts; detached forward path.
Token rates	baseline 262,144 tokens/step; C=4 H100 524,288 tokens/step; C=8 L40S/NVIDIA RTX 6000 Ada 1,048,576 tokens/step.
Optimization	AdamW-style training; learning rate $5 \cdot 10^{-4}$, warmup 2,000 steps, weight decay 0.1.
Synchronization and refresh	router synchronized every step in the tested large-run recipe; backbone and other state synchronized on slower tiered cadences; surrogate refresh/sync every 5 steps in the production recipe.
Evaluation	validation every 200 native steps using 20 validation batches.

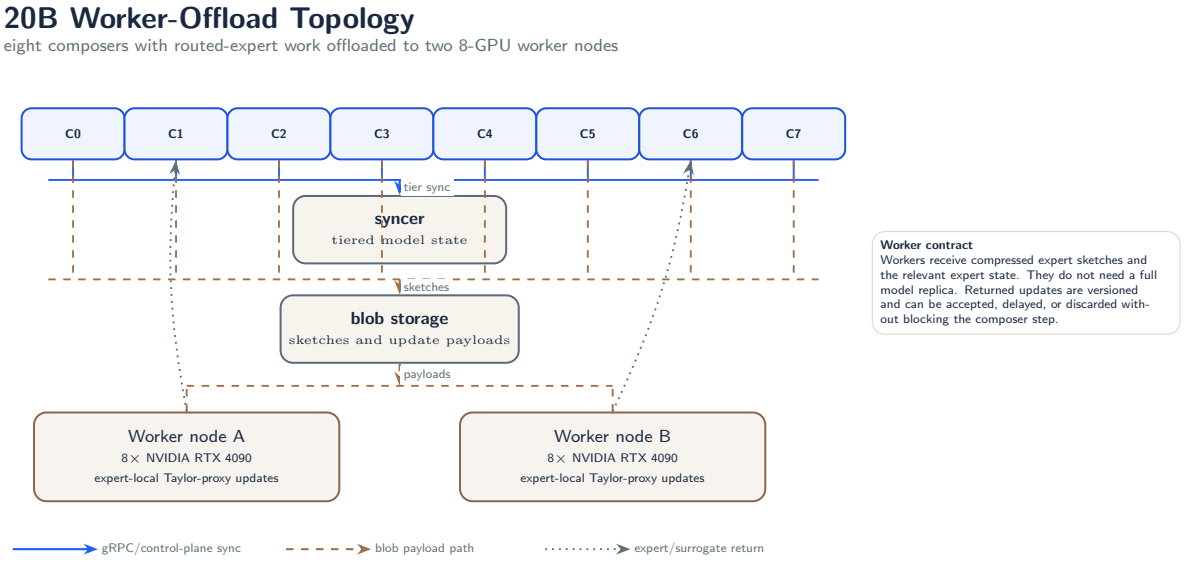

Figure 7: 20B C=8 worker-offload run topology. The generic PARALLAX architecture in Figure 1 becomes a concrete deployment with eight L40S/NVIDIA RTX 6000 Ada composers and two 8-GPU NVIDIA RTX 4090 worker nodes; composers emit compressed sketches, workers perform expert-local Taylor-proxy optimization, and versioned expert/surrogate updates return off the critical path.

For token alignment, the baseline consumes 262,144 tokens per native step. The H100 run consumes twice that amount each update, so H100 step 15,000 aligns to baseline step 30,000. The L40S/NVIDIA RTX 6000 Ada run consumes four times the baseline token count per update, so L40S/NVIDIA RTX 6000 Ada step 12,500 aligns to baseline step 50,000.

Table 6: Completed proxy-validation experiments from the deduplicated run corpus. Gaps are final validation-loss gaps relative to the paired end-to-end run.

Configuration	E2E	Parallax	Gap	Step
small ternary LatentMoE-64	4.3681	4.4837	+2.65%	10,000
small ternary LatentMoE	4.1603	4.0791	-1.95%	10,000
small ternary SwiGLU	4.1091	4.1726	+1.55%	10,000
small float LatentMoE	4.0821	4.1021	+0.49%	10,000
medium ternary LatentMoE	4.0860	4.1424	+1.38%	10,000
medium float LatentMoE	4.0659	4.0979	+0.79%	10,000

Table 7: Surrogate calibration and detached-forward ablations. All rows use the same small ternary backbone-only protocol.

Arm	Final val	Best val	Rank	Refit
backbone only real	4.4676	4.4614	32	–
backbone only surr c8 recal5	4.4851	4.4766	32	5
backbone only surr c8 recal10	4.4882	4.4802	32	10
backbone only surr c8 nodetach5	4.4907	4.4825	32	5
backbone only surr c8 rotate20 nodetach	4.6219	4.6177	32	5
backbone only surr c8 rotate20	4.6341	4.6284	32	5
backbone only surr nodetach5 r64	4.6516	4.6444	64	5

Table 8: Completed multi-composer synchronization experiments.

Setting	C	H	Final val	Points	Status
sw4 dense	6	1	4.2178	100	complete
sw streaming 05	8	32	4.3854	50	complete
sw diloco hb h64	8	64	4.5740	50	complete
sw diloco hb h128	8	128	4.7782	50	complete

6.1 Run Reconstruction and Token Alignment

The large runs use different global token rates. Native steps would misstate the comparison: 15,000 steps of one run is not the same training budget as 15,000 steps of another. Validation points are aligned by consumed tokens. If a run has global batch B, sequence length S, and one optimizer update per logged step, its token rate is $B \cdot S$ tokens per native step. The baseline token rate is 262,144 tokens/step. The four-composer H100 run uses 524,288 tokens/step. The eight-composer L40S/NVIDIA RTX 6000 Ada run uses 1,048,576 tokens/step.

Validation curves are compared by interpolation on the token axis. For a validation point at token count T, the baseline comparison value is the baseline validation curve interpolated at T, not the nearest native baseline step unless the point lands exactly on an evaluation step. Under this accounting, the C=4 H100 final point is compared to the 30k-baseline-step budget, while the C=8 L40S/NVIDIA RTX 6000 Ada 12,500-step point is compared to the 50k-baseline-step budget.

6.2 Deduplication and Cohort Pairing

Controlled experiments are paired within their experimental cohort and model configuration. A proxy arm is not compared against the best baseline observed anywhere in the corpus; it is compared against the baseline from the same cohort when such a baseline exists. This avoids overstating or understating proxy quality due to seed, launch, or configuration differences. Exact-content duplicates are collapsed before the pair selection step. When multiple same-variant pairs remain, the table reports a representative same-cohort pair with the smallest absolute final-loss gap; full robustness claims require broader sweeps.

6.3 Interpretation of Negative and Positive Gaps

Some small controlled comparisons finish with a negative proxy gap, meaning the proxy arm has lower final validation loss than its paired end-to-end baseline. The paper does not claim proxy training is inherently superior to end-to-end

training. These cases show that the proxy path can be competitive within run noise and local hyperparameter variation. Across several settings, proxy/offloaded expert training tracks end-to-end training closely enough to support the large-scale decomposition.

6.4 Evaluation Granularity

The 20B results use median composer validation curves for the multi-composer runs. Median aggregation is appropriate because composers are symmetric samples of the same training process but may have transient local noise from heterogeneous hardware, network timing, and worker scheduling. Interquartile bands in Figure 8 show the spread across composers where available. The baseline is a single end-to-end run and is therefore plotted without a composer band.

7 20B Results

Table 9: 20B token-matched validation results. Gaps are relative validation-loss gaps; lower is better.

Run	Native step	Base-eq. step	Val loss	Gap
B300 E2E baseline	50,000	50,000	3.1475	0.00%
Parallax C=4 H100	15,000	30,000	3.1896	-0.29% at matched budget
Parallax C=8 L40S/NVIDIA RTX 6000 Ada, 50k-equiv.	12,500	50,000	3.2521	+3.32%
Parallax C=8 L40S/NVIDIA RTX 6000 Ada, final	15,200	60,800	3.2275	+2.54% vs baseline final
Parallax C=8 L40S/NVIDIA RTX 6000 Ada, best median	14,700	58,800	3.2104	+2.00% vs baseline final

Table note. The C=8 final and best-median rows extend beyond the baseline's exported 50k-step token budget. Their gaps use the baseline final validation loss because no longer-baseline point is available.

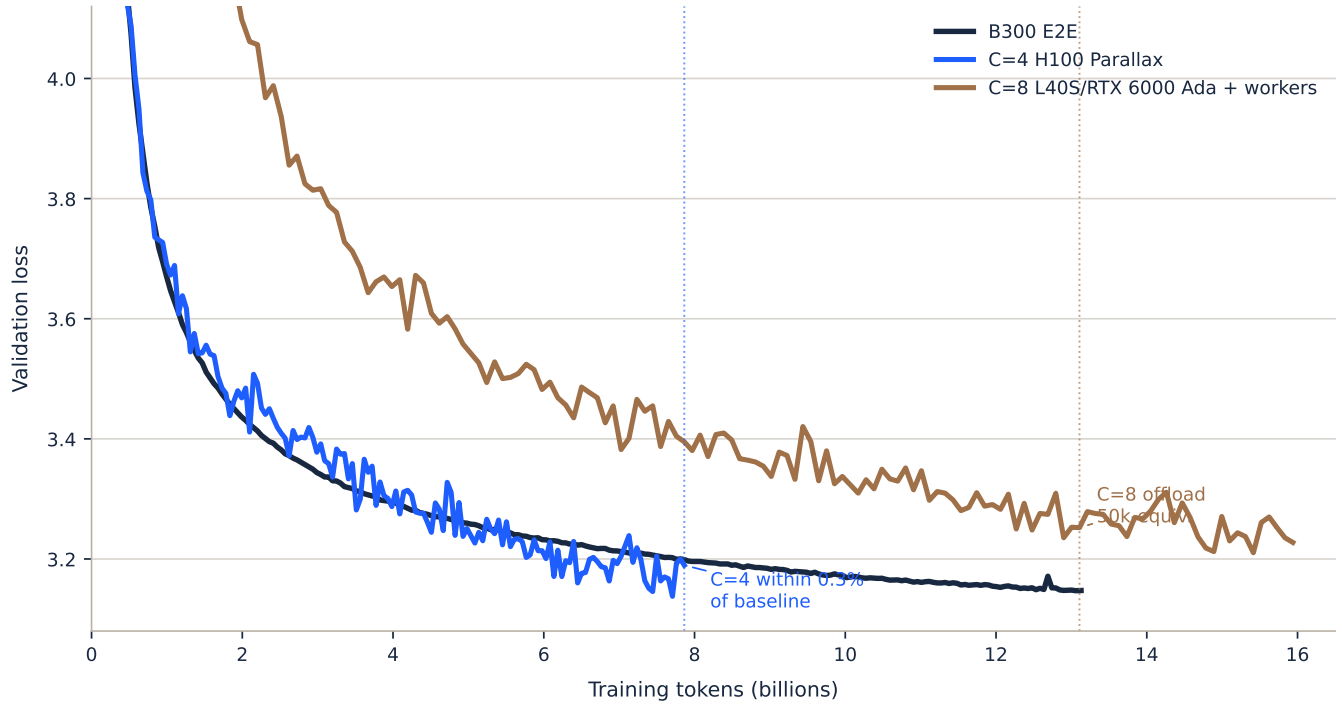

Figure 8: 20B validation loss aligned by consumed tokens. The four-composer H100 run is within 0.3% of the B300 baseline at the 30k-equivalent comparison point. The C=8 L40S/NVIDIA RTX 6000 Ada worker-offload run reaches +3.32% at 50k baseline-equivalent steps and continues improving after that point.

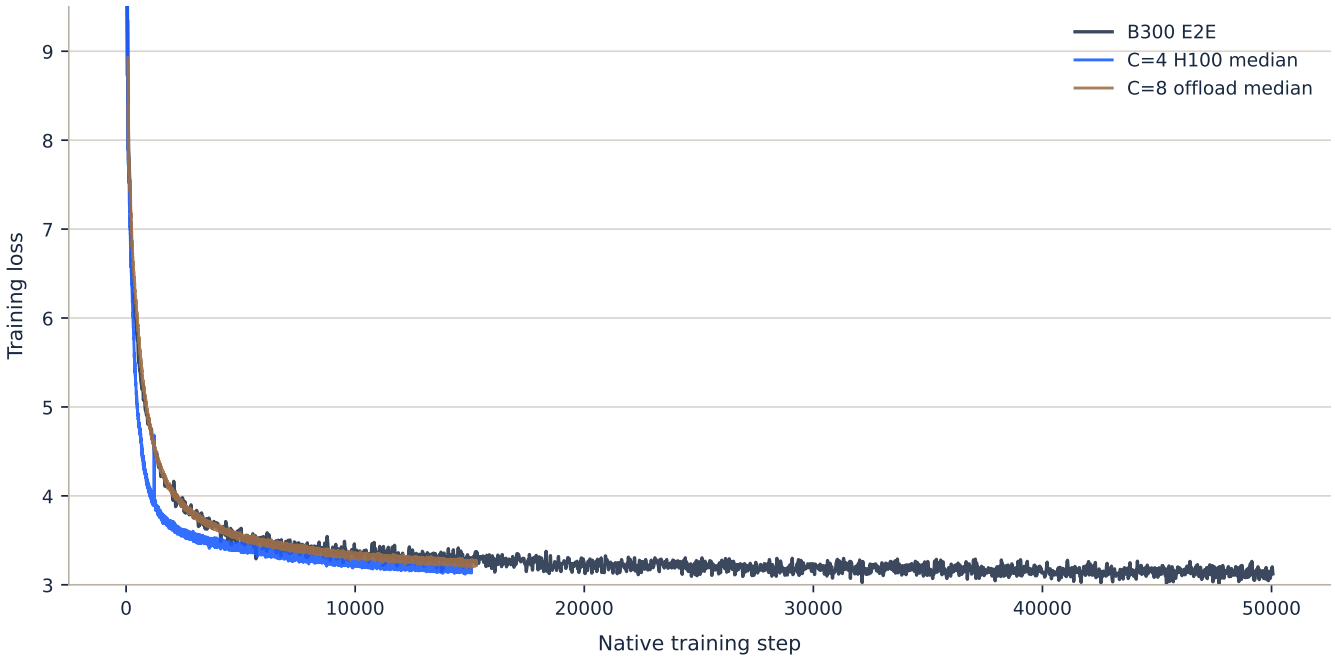

Figure 9: 20B training loss on native step axes. This view is not token-normalized; it shows run continuity and the distinct native step scales of the baseline, C=4, and C=8 runs.

The C=4 run is the cleanest quality comparison in the report. At the matched 30k-equivalent point, the median PARALLAX validation loss is 3.1896 and the interpolated baseline value is 3.1990. This is a point estimate at equal consumed tokens; no replicated equivalence test is reported.

The C=8 worker-offload run is the harder systems setting. It uses more composers, cheaper composer GPUs, and remote expert workers. It is also noisier. At the 50k-equivalent point it is +3.32% versus the baseline. It continues to improve after that point and reaches a best exported median of 3.2104, +2.00% versus the baseline final value. The run does not prove the offload implementation is throughput-optimal; it shows that a 20B model can keep training while routed-expert optimizer work is moved off the composers.

7.1 Quality Versus Systems Stress

The two PARALLAX arms are intentionally different. The H100 run stays close to the direct ownership model: four composers, owned-expert training on the composer, no worker fleet. The L40S/NVIDIA RTX 6000 Ada arm pushes the offload path: more composers, cheaper GPUs, remote workers, and more asynchronous noise. The cost model explains the trade: at C=8, direct ownership gives 7.6× on routed experts with 2.16× total active-compute reduction; offload gives 20.9× on composer-side routed experts with 2.43× total active-compute reduction.

7.2 Baseline Strength

The baseline uses the matched 20B model family trained end-to-end on four B300 GPUs. It reaches validation loss 3.1475 at the 50k-step point and is already near the relevant token budget for the active parameter count. We use it as a strong quality comparator rather than as a communication-bound baseline.

7.3 Scope of the 20B Evidence

The 20B experiments do not prove that arbitrary composer counts work, that any surrogate rank is sufficient, or that worker offload is already throughput-optimal. They also do not isolate every architecture choice. The demonstrated setting is narrower: a 20B LatentMoE model with ternary routed experts, hard expert ownership, low-rank non-owned surrogates, and tiered synchronization.

8 Controlled Validation

The 20B runs combine the pieces. The controlled experiments pull them apart. Table 6 reports completed same-cohort proxy-validation experiments. The medium ternary run finishes 1.38% above its end-to-end counterpart, and the medium float run finishes 0.79% above. Small variants show similarly tight gaps, including a SwiGLU ternary run at 1.55% and a float LatentMoE run at 0.49%. A small ternary LatentMoE comparison slightly favors PARALLAX; the paper treats it as a competitive proxy-training result. It is not used as a superiority claim.

Table 10: Representative Taylor-proxy worker-offload runs. Proxy hours count expert-local worker optimization time; updates are expert-update applications accumulated during training.

Run	Step	Val	Updates	Updates/s
medium float Parallax rank 8 Adam	10,000	4.0979	3,563,527	43.78
medium ternary Parallax rank 8 Adam	10,000	4.1424	3,591,486	46.91
small ternary Parallax C=4 worker rank 8 Adam matched LR	10,000	4.1025	1,108,659	22.32
small ternary Parallax C=4 worker rank 8 Adam expert LR low	10,000	4.0791	1,067,564	22.68
small ternary Parallax C=4 worker rank 8 Adam expert LR high	10,000	4.1858	1,111,596	22.44
small ternary Parallax C=8 worker rank 8 Adam	10,000	4.1309	543,398	11.08
small ternary Parallax C=16 Adam worker	10,000	4.2923	272,806	30.54

Table 11: Surrogate forward calibration in the backbone-only protocol. The best C=8 settings keep surrogate validation predictions within sub-percent mean absolute error while preserving low-rank local execution.

Run	Real val	Surr. val	Mean abs. gap	Max abs. gap
real experts	4.4614	4.4614	0.000%	0.000%
surrogate C=8 recal5	4.4766	4.4690	0.084%	0.178%
surrogate C=8 recal10	4.4802	4.4772	0.033%	0.089%
surrogate C=8 nodetach5	4.4825	4.4760	0.062%	0.155%
surrogate C=8 rotate20	4.6284	4.6121	0.469%	1.140%
surrogate C=8 rotate20 nodetach	4.6177	4.6021	0.299%	0.906%
surrogate nodetach5 r64	4.6444	4.6076	0.374%	1.199%

Table 12: Router synchronization telemetry. Higher top-k overlap and lower router KL indicate that composers retain a shared routing interface.

Run	Mean overlap	Final overlap	Mean KL	Final KL
Dense sync	0.954	0.956	0.0043	0.0040
Heavy-ball H=64	0.576	0.463	0.2806	0.2603
Heavy-ball H=128	0.542	0.454	0.4587	0.3947
Streaming alpha=0.5	0.610	0.516	0.3094	0.4181

Table 13: Batched Taylor-proxy diagnostic. Batched worker optimization matches the 250-step validation loss of sequential processing while reducing wall-clock step time.

Mode	Last step	Val	Mean ms/step	Updates
Sequential proxy	250	7.1260	6963	26,050
Batched proxy	250	7.1271	2358	25,986
Speedup	–	–	2.95×	–

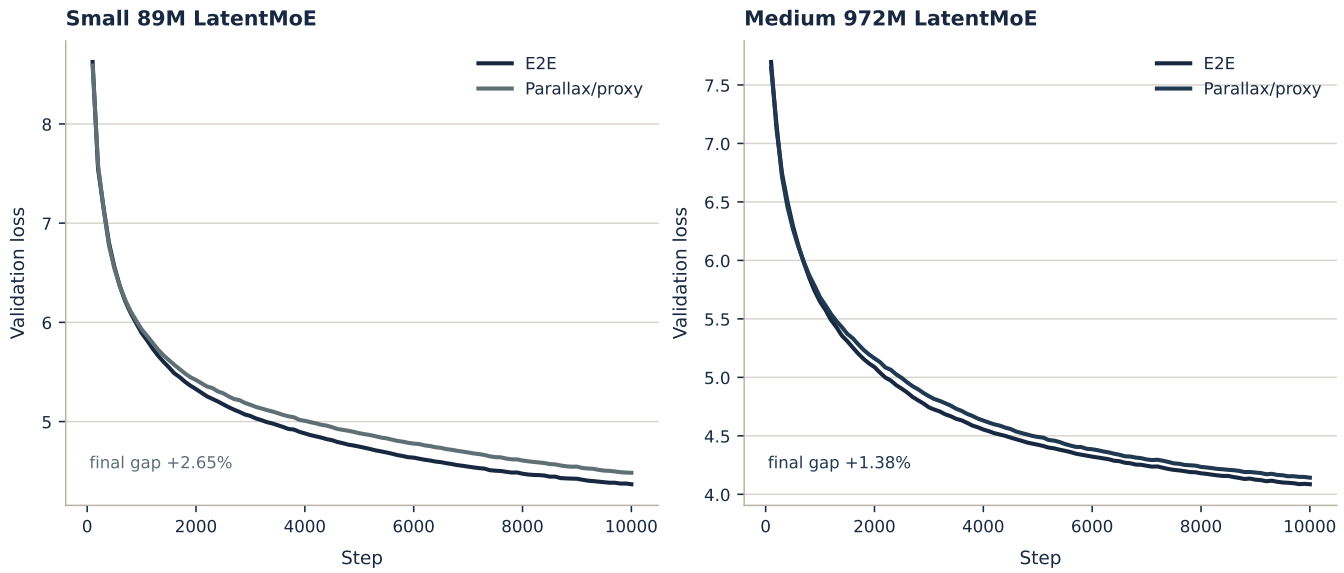

Figure 10: Representative proxy-validation curves at small and medium scale. Proxy/offloaded expert training tracks end-to-end training when paired within the same completed-run cohort.

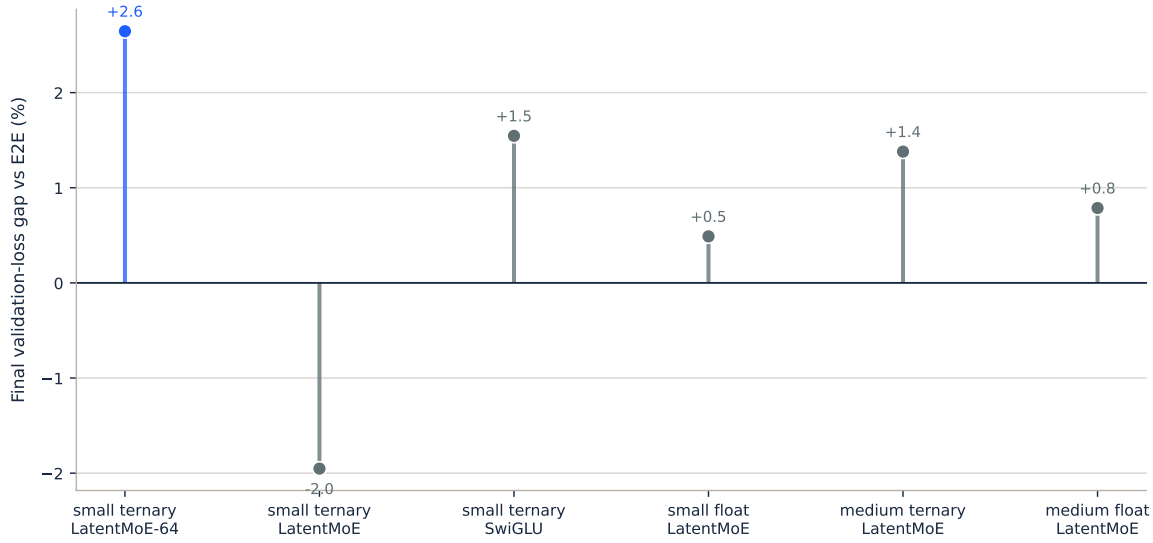

Figure 11: Final validation-loss gaps for completed proxy-validation pairs. Negative values mean the proxy arm finished lower than its same-cohort end-to-end baseline.

Table 7 and Figures 12–13 examine surrogate quality. The C=8 detached surrogate calibration rows with frequent refits land close to the real-expert reference in a backbone-only protocol. That is the regime surrogates must satisfy: useful forward context without a non-owner expert backward pass.

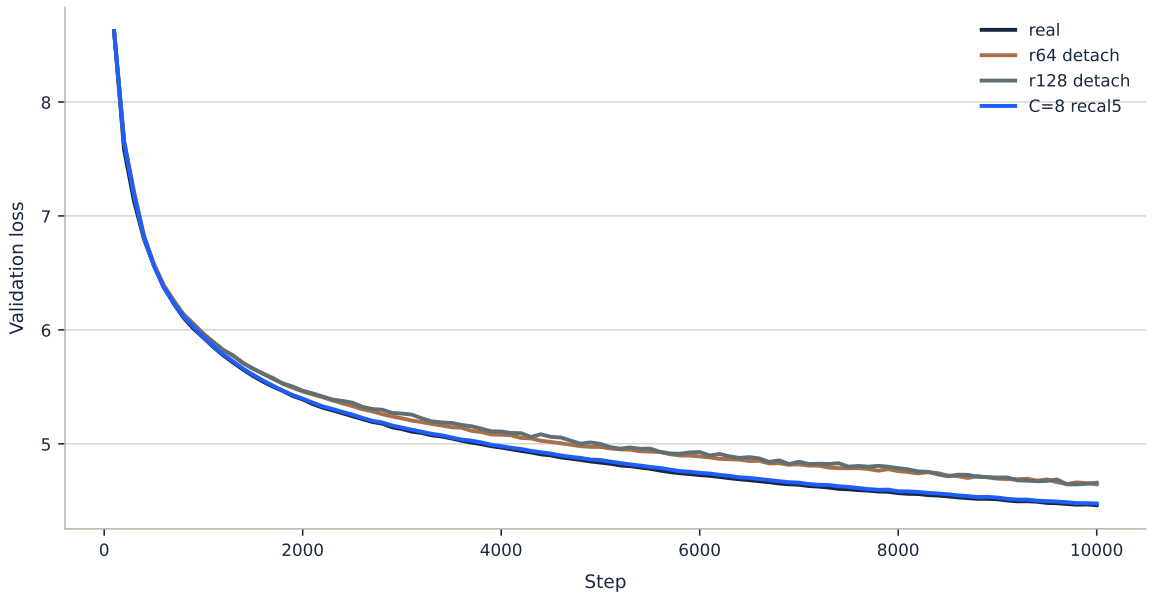

Figure 12: Surrogate calibration curves. Low-rank surrogates track real expert behavior closely enough to support backbone/router training while avoiding non-owned expert backward passes.

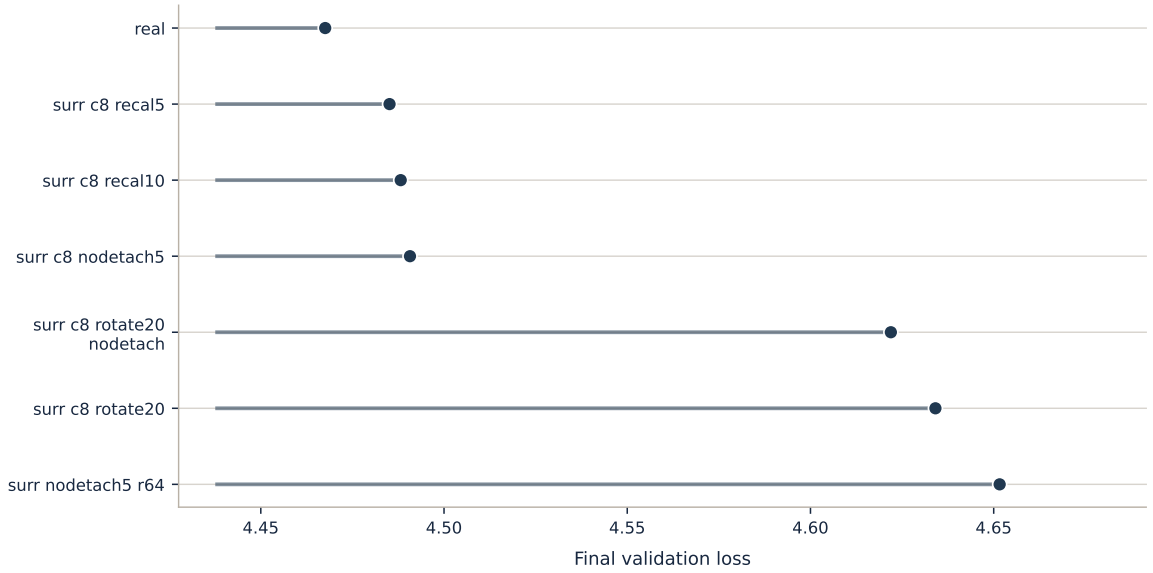

Figure 13: Final validation loss across surrogate ablations in the completed backbone-only protocol.

The multi-composer synchronization table reports completed settings that train successfully under several synchronization cadences. Dense or hierarchical backbone synchronization works at moderate horizons in these runs. No single horizon is presented as universal; the expert decomposition leaves a smaller shared-parameter synchronization problem that can be tuned independently of expert all-to-all.

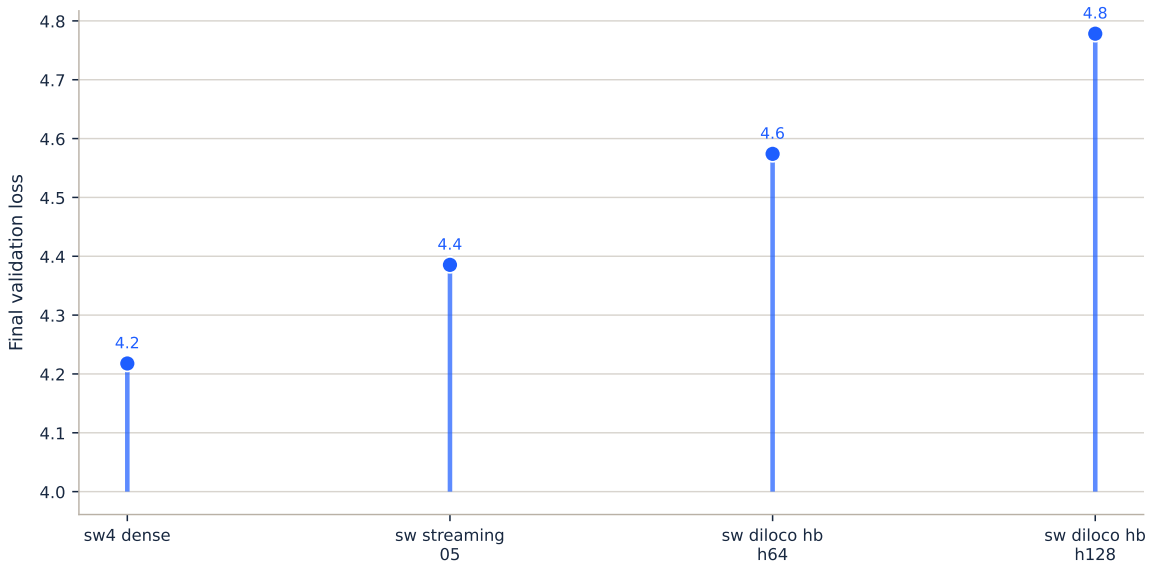

Figure 14: Audited multi-composer synchronization outcomes for completed settings.

8.1 Proxy Validation Across Precision and Architecture

The proxy-validation table includes both ternary and float experts. Ternary experts could in principle make the proxy easier by reducing the effective degrees of freedom, or harder by introducing quantization-induced discontinuities in the forward representation. The observed gaps remain small in both categories. The medium float comparison is within 0.79%, and the medium ternary comparison is within 1.38%. This suggests the proxy path works across more than one quantization regime.

The small SwiGLU result is also informative. The main large models use relu^2 routed experts, but the method is not tied to that activation. A SwiGLU expert has a different local curvature and parameterization, yet the proxy pair remains close. The broader conclusion is that Parallax depends on a compact expert interface and a usable local proxy objective rather than one exact MLP nonlinearity.

8.2 Taylor-Proxy Worker Evidence

Table 10 reports representative worker-offload runs in which routed-expert updates are produced through Taylor-proxy objectives rather than ordinary composer-side expert backpropagation. The update volume is substantial: medium-scale runs apply more than 3.5M expert updates over 10k steps, while small C=4/C=8/C=16 runs exercise the same mechanism under different ownership fractions. These runs show that Taylor-proxy worker updates can be sustained over 10k-step small and medium runs while preserving close validation-loss tracking.

The worker objective is intentionally local. It uses the captured expert input and output-gradient relation from the real training step, keeps the nonlinear expert forward under candidate weights, and bounds the update with a proximal term. This motivates the offload path over pure black-box or stale-sketch alternatives: the worker optimizes the part of the loss that an expert can directly affect.

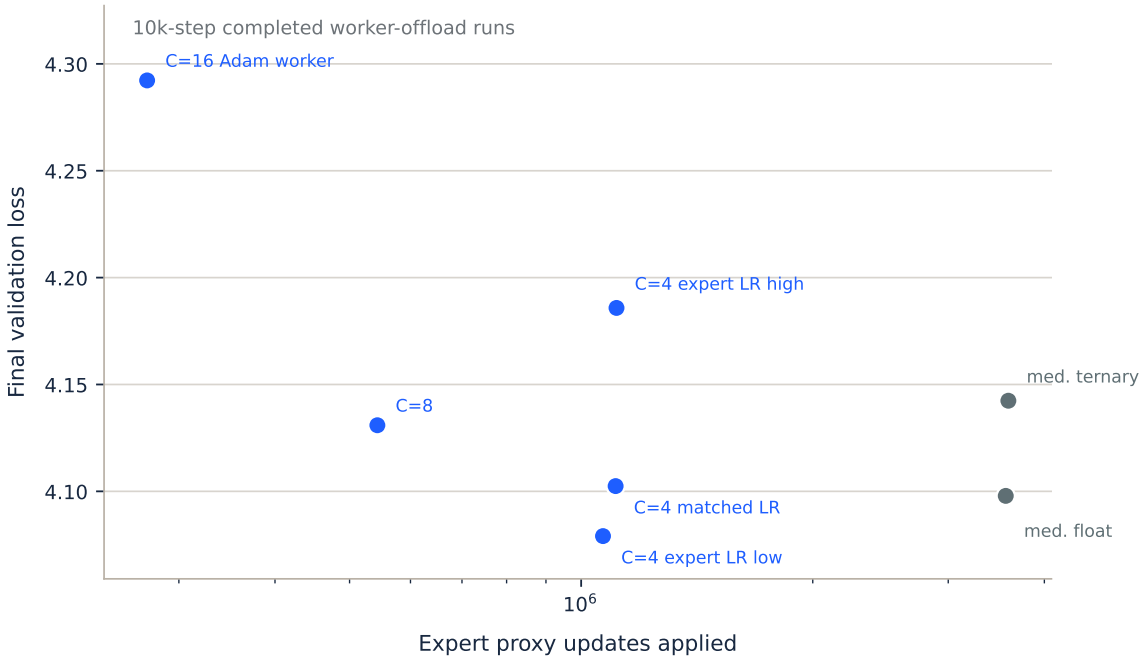

Figure 15: Representative Taylor-proxy runs by expert-update count and final validation loss. The figure emphasizes that worker updates are sustained across small and medium settings rather than used only for short diagnostics.

Table 13 isolates a practical worker implementation detail. At the shared 250-step horizon, batched proxy processing matches sequential proxy validation loss while reducing mean step time by 2.95×. A mature offload implementation should process multiple sketches per expert update and compress once per return path, rather than treating every sketch as a separate synchronous unit.

8.3 LatentMoE Ablations

LatentMoE compresses the expert interface. Controlled runs include latent and no-latent variants. The no-latent variants are useful as architecture comparisons, but they are less attractive for the offload path because sketch size, surrogate size, and expert all-to-all accounting scale with the expert interface dimension. A no-latent expert can be trained with the same ownership idea, but it weakens the systems argument. For large models, LatentMoE is the more favorable design point because the same dimension d_e governs expert arithmetic, sketch bandwidth, surrogate storage, and routing interface cost.

8.4 Surrogate Rank and Refresh Cadence

Surrogate quality is controlled by rank, refresh cadence, and the data used for recalibration. A larger rank reduces approximation bias but increases memory and surrogate forward cost. A more frequent refresh reduces staleness but increases background traffic and owner work. Table 7 shows that C=8 surrogate calibration with refresh every 5–10 steps is close to the real-expert reference in the backbone-only protocol. The rotated or less direct surrogate settings are worse, so the default remains a simple local low-rank surrogate.

Table 11 measures surrogate forward prediction directly. The strongest C=8 refresh settings have mean absolute surrogate-validation gaps of 0.033–0.084% and maximum gaps below 0.18%. This does not settle every staleness regime, but it explains why low-rank non-owned forward calls are viable in the tested recipe.

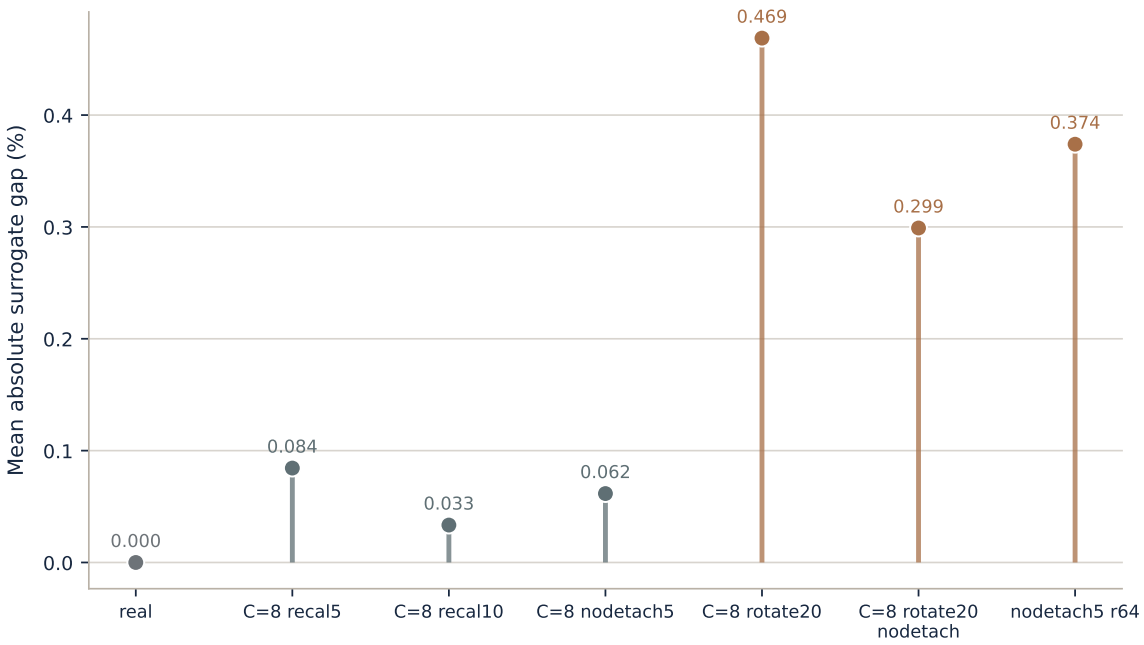

Figure 16: Mean absolute surrogate-validation gap in the backbone-only surrogate protocol. Frequent C=8 refresh keeps the surrogate forward path tightly calibrated.

8.5 Synchronization Sweep Interpretation

The multi-composer rows show a predictable pattern: frequent dense synchronization is strongest, while longer horizons trade quality for lower communication. Several nontrivial multi-composer settings train successfully. PARALLAX does not make all outer optimizers interchangeable; it reduces the remaining synchronization problem enough to tune it separately from expert execution.

Table 12 and Figure 17 add routing-specific telemetry. The dense synchronization setting preserves high top-k overlap and low router KL across composers, while slower synchronization settings lose shared routing semantics. Expert ownership prevents destructive expert averaging; frequent router/latent synchronization keeps the namespace shared.

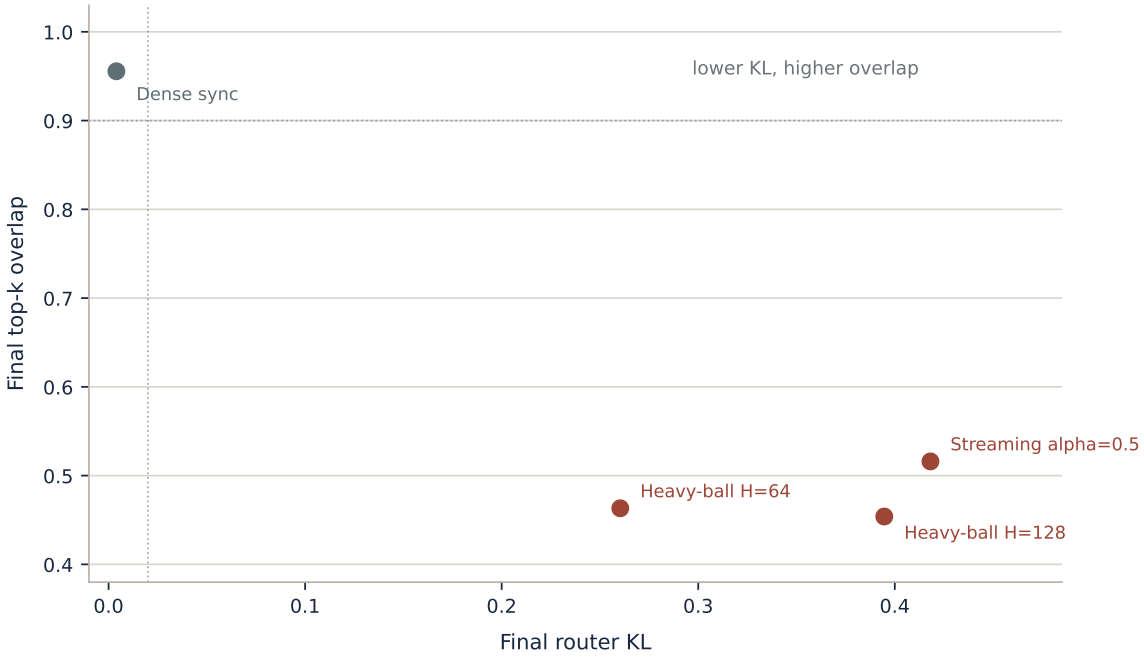

Figure 17: Router telemetry from controlled multi-composer runs. Tighter router synchronization preserves a shared routing interface much more effectively than slower alternatives.

8.6 Role of Controlled Experiments

Small and medium experiments cannot prove 20B behavior, but they isolate effects that the 20B runs combine. A 20B training curve cannot separately answer whether a surrogate rank is adequate, whether proxy updates are locally useful, or whether a sync horizon is stable. The controlled corpus supplies those answers under cheap repeatable conditions. The large runs then validate that the combined recipe still works under realistic model size, hardware constraints, and network behavior.

9 176B Feasibility and Scaling

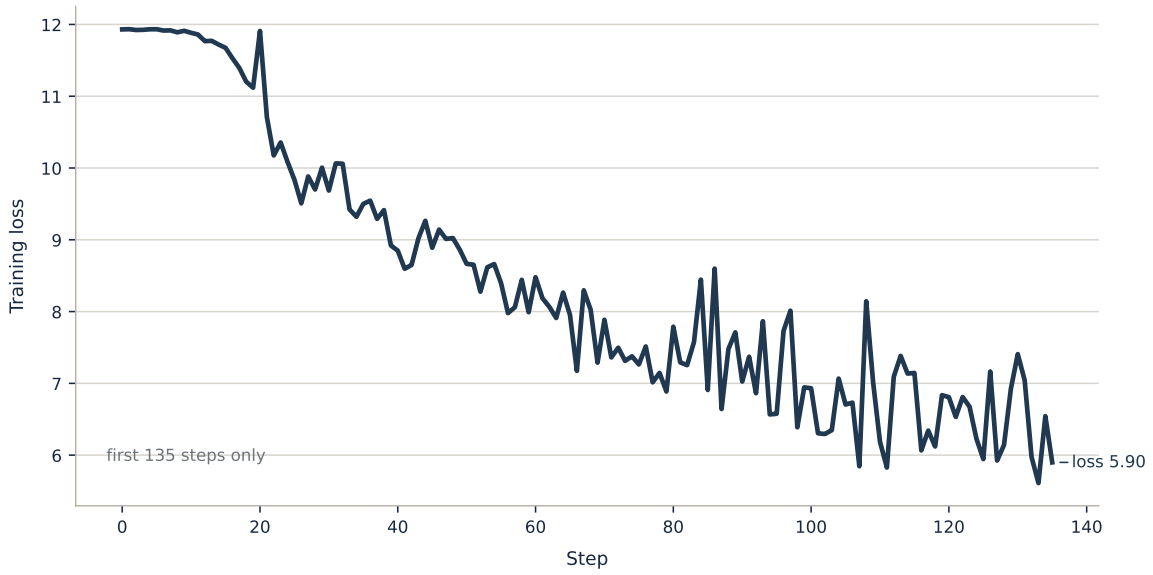

Figure 18: 176B feasibility smoke test, truncated to the first 135 steps. This is systems feasibility evidence only: architecture initialization, expert-namespace routing, wide-area synchronization, and early loss reduction across four B300 nodes.

The 176B run is a feasibility smoke test. It does not report convergence. It exercises the large-model path we care about: sparse ternary experts for capacity, LatentMoE for a compact expert interface, a Mamba-heavy backbone for lower decode-state pressure, a small attention budget for binding and retrieval, and manifold-constrained hyper-connections for deep residual transport. Those choices are not needed to prove the 20B training result. They define an inference-oriented target architecture, and the first 135 steps show that the target can initialize, route over its expert namespace, synchronize across four nodes, and reduce training loss.

Nemotron-style hybrid Mamba-attention MoE releases are useful context for that target. They should not be read as evidence for PARALLAX training. NVIDIA describes Nemotron 3 Ultra as a 550B total-parameter open model with up to 55B active parameters per token, hybrid Mamba-Attention blocks, and sparse MoE structure [NVIDIA, 2025].[1][2] The relevance is architectural: efficient long-context inference increasingly points toward sparse capacity, compact expert interfaces, low-bit expert storage, and attention/state-space hybrids. PARALLAX is a training-side attempt to make that class of model easier to build when compute is fragmented.

10 Limitations

The completed large experiments test up to $C = 8$ composers. Larger composer counts remain extrapolation. The cost model shows favorable scaling with C, but optimizer noise, surrogate freshness, router drift, and worker reliability may become limiting factors before the theoretical savings are fully realized.

The worker-offload implementation is not yet optimized. Sketch capture, compression, batching, worker scheduling, surrogate recalibration, and update application all have engineering headroom. The L40S/NVIDIA RTX 6000 Ada result should be read as a quality result under offload, with throughput benchmarking deferred until the implementation is more optimized.

The paper reports hard ownership. It does not yet validate replicated ownership, where multiple composers own the same expert shard and average or RDA-DiLoCo their real expert states. Replicated ownership is likely useful for $C = 16$ and above because it reduces owner-specific noise and improves fault tolerance.

The 176B result is a smoke test. It supports launch-path feasibility only. Full validation requires a long run, clean eval cadence, and architecture-specific ablations.

[1] https://docs.nvidia.com/nemotron/nightly/usage-cookbook/Nemotron-3-Ultra-Base/README.html
[2] https://research.nvidia.com/labs/nemotron/Nemotron-3-Ultra/

11 Future Work

Replicated expert shards. The natural next experiment is $C = 16$ or $C = 32$ on small and medium models while keeping the number of owned expert shards fixed. For example, 32 composers could map onto 8 expert shards with four replicas per shard. Co-owners would periodically consolidate real expert weights, optimizer moments, and surrogate calibrations. The consolidation rule should be tested directly: simple averaging, RDA-DiLoCo, outer momentum, and per-expert trust-region averaging are all plausible.

Surrogate-update protocols. The present surrogate protocol is conservative. Future versions should compare full refit, low-rank delta broadcast, moment-matched surrogate updates, error-feedback residuals, and rank-adaptive updates. The key metric is not surrogate reconstruction error alone, but downstream validation loss at fixed compute and communication budgets.

Worker fleets and sketch compression. The offload path should be benchmarked as its own system. Important axes include sketch precision, HQ4 compression, deduplication ratio, worker batch size, Taylor-proxy horizon, stale-sketch tolerance, and expert update acceptance rules. A mature version should let composers train without routed-expert Adam state while cheap worker GPUs join and leave without destabilizing the run.

Inference-oriented architectures. The next large runs should keep training and serving costs in the same frame. The 176B smoke-test architecture is one plausible target: ternary routed experts for cheap capacity, LatentMoE for compact sketches and surrogates, a Mamba-dominant backbone for lower long-context decode cost, selective attention for retrieval and binding, and manifold-constrained hyper-connections for stable depth [Gu and Dao, 2023, Dao and Gu, 2024, Elango et al., 2026, Wang et al., 2023, Ma et al., 2024, Lidin et al., 2026]. This exact architecture is not proven. It is a good inference-oriented stress case for Parallax because it combines the same properties the training method tries to exploit: sparse experts, compact expert interfaces, and a large gap between total and active capacity.

Surrogate-only expert inference. The training results raise a sharper inference question: if low-rank surrogates provide enough forward signal for non-owned routed experts during training, can some or all routed experts be collapsed into their surrogates for final inference without a material quality loss? This would turn the surrogate bank from a training device into an inference compression path. The experiment should evaluate full-expert, mixed full/surrogate, and all-surrogate expert serving at matched checkpoints, measuring validation loss, downstream task quality, latency, memory bandwidth, and expert-cache behavior. A positive result would make Parallax more than a distributed training method: the same decomposition that removes remote expert execution during training could also reduce expert serving cost after training.

Serving evaluation. Future Parallax checkpoints should be judged as inference systems as well as training runs. The required measurements are validation loss, downstream task quality, prefill tokens/sec, decode tokens/sec, KV-cache memory, active expert bandwidth, expert-cache behavior, long-context throughput, and cost per generated token. Nemotron-style systems are a useful reference point because they package sparse MoE, hybrid Mamba/attention blocks, long-context operation, and low-precision deployment in one serving-oriented model family. They do not validate Parallax; they help define the bar.

Large-scale validation. The 176B design uses roughly 176B total parameters with about 8.7B active parameters, 32 MoE layers, 640 experts per layer, top-$k = 16$, latent dimension 1024, and larger 1024→4096→1024 relu^2 experts. The intended validation sequence is to reproduce the 20B result with better worker scheduling, close the targeted surrogate and worker-offload gaps in controlled runs, and then train the 176B architecture long enough to measure both convergence and inference performance.

12 Design Search and Negative Results

The method is narrow because the search space is large. Many plausible approaches appear simpler than Parallax but fail one of the core requirements: local completeness, expert-state locality, asynchronous expert work, or tiered synchronization. The negative results explain the shape of the design rather than acting as filler ablations.

12.1 Pure Evolution Strategies

One natural way to avoid backpropagating through remote experts is to use black-box evolution-style perturbations of expert weights. This has an appealing systems property: workers can evaluate perturbations locally and return scalar fitness estimates or signed updates. In practice, the signal-to-noise ratio is too poor for the expert sizes and update cadence required here. Expert MLPs are high-dimensional, and the routed-token subset for one expert provides a narrow local view of the global objective. Pure ES spends too much compute estimating a direction that activation sketches already provide through first-order output gradients.

The lesson is that worker independence is not enough. Workers need a useful local training signal. Taylor-proxy optimization keeps the worker problem local while preserving first-order information from the real training step.

12.2 One-Sided Low-Rank Adaptation

Another tempting design is to update only one side of the expert MLP or represent expert updates as a small LoRA adapter. This reduces communication, but it also constrains the update geometry too aggressively. Routed experts need to change both input-to-hidden and hidden-to-output mappings. A one-sided update can fit narrow sketch objectives but does not provide a general enough correction to keep long training stable.

Low rank is used for non-owned recall. Real owned experts and worker-updated experts remain full experts.

12.3 FedProx and Generic Federated Averaging

FedAvg and FedProx-style methods address client drift by averaging full model states or adding a proximal penalty to local objectives. They are attractive because they are simple and well studied. They are poorly matched to sparse MoE expert ownership. Averaging whole expert sets destroys the ownership advantage; each participant must still hold or exchange too much expert state. Applying a generic proximal term also treats all parameters as if they had the same synchronization tolerance, which is exactly what the MoE structure contradicts.

PARALLAX keeps the proximal/trust-region idea where it fits: worker expert updates are bounded by a local Taylor proxy and versioned acceptance checks. It does not use a full-model federated objective as the primary decomposition.

12.4 Averaging Routed Experts Across Private Routers

A more MoE-specific variant is to let every island train the same expert ids locally and periodically average only the routed experts. This preserves some sparse structure, but it fails to respect the router/expert contract. Expert id 17 on one island may receive code-heavy tokens while expert id 17 on another receives math-heavy or multilingual tokens. Averaging the two experts is not equivalent to training one better generalist; it can erase the very specialization sparse routing is supposed to create.

PARALLAX avoids this semantic collision by making expert ownership explicit. Expert ids have authoritative update streams. Non-owners retain visibility through surrogates, and routers are synchronized frequently enough that the namespace remains shared. This is a narrower synchronization target than full expert averaging and a better match to how MoE capacity is used.

12.5 Sequential Deflation

Sequentially training or refreshing expert components one at a time can reduce peak memory and isolate expert updates. The cost is stale interaction with the router and backbone. MoE training couples the router, latent interface, and expert outputs, so treating experts as independent regressions loses useful co-adaptation. Sequential deflation makes the stale-state problem worse by delaying some experts too long and overfitting others to narrow slices of training context.

PARALLAX uses concurrent ownership and asynchronous worker updates instead. Experts can update independently, but the global training loop continues to present fresh router and backbone context.

12.6 GaLore-Style Projection for Expert Updates

Low-rank gradient projection is useful when gradients are dense and communication is the bottleneck. Expert offload has a different bottleneck. The worker already operates on a small expert-local problem, and the update must survive compression and asynchronous application. A global low-rank projection basis can suppress useful sparse expert-specific corrections. It also adds another synchronization object that must remain consistent across composers and workers.

PARALLAX uses low-rank structure for surrogate recall and optional surrogate deltas, where the approximated object is explicitly low rank. It does not force every real expert update through a shared low-rank gradient subspace.

12.7 Shared SVD Bases

A shared SVD basis across experts is appealing because it amortizes communication: send coefficients instead of full matrices. The problem is specialization. Routed experts are supposed to diverge into different functions. A shared basis can be efficient early in training but becomes a bottleneck as experts specialize. It also creates a fragile global object whose update cadence is unclear: synchronize it too often and communication returns; synchronize it too rarely and the basis becomes stale.

PARALLAX instead lets each expert owner recalibrate that expert's surrogate. The approximation basis is local to the expert rather than imposed globally across the layer.

12.8 Unbiased Random Projections

Random projection methods can preserve inner products in expectation and are attractive for sketching. Unbiasedness alone is insufficient. The worker update must remain useful after finite-sample sketching, compressed transport, version lag, and application to a nonconvex expert. Unbiased estimators with high variance are less valuable than biased but stable local objectives. The Taylor sketch preserves the actual expert input/output-gradient relation, which is the information the worker needs most.

12.9 L-BFGS Inner Optimization

Quasi-Newton inner loops can reduce the raw worker proxy loss quickly. That is not the same as producing a good compressed update. A broad low-magnitude quasi-Newton update can be damaged by top-k sparsification or low-bit compression, while an Adam-style update may concentrate mass in coordinates that survive the return path. The relevant objective is improvement after compression and acceptance.

This motivates compression-aware worker optimization. Future worker optimizers should be evaluated after the exact compression and acceptance pipeline, including the uncompressed local objective only as a diagnostic.

12.10 Historical Sketch Banks

Maintaining a large bank of old sketches appears to improve data diversity, but it introduces stale output gradients. The expert input distribution, router weights, and upstream gradients all move during training. A worker optimizing against old sketches can make locally plausible updates that are misaligned with the current model. Small recency windows may be useful, but large historical banks are a poor substitute for fresh sketch diversity.

The offload design prefers fresh multi-batch sketches over deep optimization on stale cached data. Asynchronous work is useful only while the local objective remains close enough to the current model.

12.11 Full Expert Replication

The simplest way to avoid remote expert dispatch is to replicate every expert on every composer. This gives local completeness but loses the memory and optimizer-state advantage. It also becomes increasingly unattractive as total expert count grows. Full replication is therefore a useful conceptual baseline rather than a viable answer to the internet-scale training problem.

The diagnostic all-expert variants make this concrete. When each composer owns the complete expert set, the optimizer footprint rises immediately; dense and top-k synchronized all-expert variants exhausted 32GB-class GPUs within tens of steps in the small multi-composer setup. This failure occurs before considering the semantic problem of averaging experts trained behind different routers. Full replication solves locality by giving up the reason to shard the MoE in the first place.

PARALLAX keeps local completeness through surrogates rather than full replication. The composer can execute every routed expert interface, but only owned experts require full parameters.

12.12 Synchronous Expert Parallelism Over Wide-Area Links

One could attempt to preserve ordinary expert parallelism and simply run it over the internet. The balanced-routing byte counts rule this out for the tested 20B settings: hundreds of gigabytes must move on the critical path each optimizer step even before overheads. The issue is step latency and jitter as well as bandwidth cost. A training step becomes fragile when every MoE layer waits for remote expert activations over commodity links.

PARALLAX removes token-level expert traffic from the critical path entirely. Ordinary expert parallelism does not.

12.13 Summary of the Design Search

The alternatives fail for different reasons, but the pattern is consistent. Methods that synchronize full models or full expert sets lose the systems advantage. Methods that make workers too independent lose training signal. Methods that overcompress real expert updates lose update quality. Methods that use old sketches lose alignment with the current model. PARALLAX occupies the middle ground: local complete execution through surrogates, real first-order sketch signal for workers, authoritative expert ownership, and tiered synchronization for the parameters that remain global.

13 Appendix: Analytical Details

13.1 Expert FLOP Derivation

For one token assigned to one relu^2 expert, the first matrix multiply costs $2d_eh$ FLOPs and the second costs $2hd_e$ FLOPs, giving $4d_eh$ forward FLOPs. Backpropagating through the two linear layers and computing weight gradients adds approximately twice the forward cost, so training cost is approximated as $12d_eh$. This constant-factor model is standard for transformer/MoE accounting and is sufficient for comparing real expert calls with surrogate calls of the same shape.

The surrogate replaces h with rank r, so forward cost is $4d_er$. Because non-owned surrogates are detached, they do not incur backward or weight-gradient cost on the composer. The direct relative expert compute is therefore

$$\rho_{\text{direct}}(C) = \frac{1}{C} \cdot 1 + \left(1 - \frac{1}{C}\right) \cdot \frac{4d_er}{12d_eh}.$$

Canceling $4d_e$ gives the expression in the main text.

13.2 Active Parameter Accounting

For L_{moe} MoE layers, k selected experts per token, and expert parameter count $2d_eh$, the active expert parameters per token are

$$P_{\text{active},E} = L_{\text{moe}}k(2d_eh).$$

The active model parameters are $P_B + P_{\text{active},E}$, where P_B includes embeddings, attention/state-space blocks, router and latent projections, normalization layers, and other always-active parameters. Total model parameters are $P_B + L_{\text{moe}}N(2d_eh)$, where N is the number of routed experts per MoE layer.

PARALLAX reduces routed-expert training compute but does not make the backbone free. Total active compute savings are bounded by the backbone share. Expert-only savings can be large while total active savings are smaller.

13.3 All-to-All Accounting

An expert-parallel MoE layer must send expert inputs to remote expert owners and receive outputs back. With global batch B, sequence length S, top-k routing, latent expert width d_e, and bf16 activations, the balanced-routing remote-token accounting per MoE layer is

$$2BSk\left(1 - \frac{1}{C}\right) d_e \cdot 2 \text{ bytes}.$$

Multiplying by L_{moe} yields the accounting expression used in the main text. Practical implementations add routing metadata, capacity padding, load-imbalance overhead, collective protocol overhead, and often extra buffering; realized traffic can also differ if routing is not balanced across remote and local expert shards.

13.4 Surrogate Sync Bound

A rank-q update to one surrogate matrix of shape $m \times n$ can be represented by factors with $q(m+n)$ scalars. A two-matrix surrogate update costs approximately $2q(d_e + r)$ scalars per expert. If each of C composers owns $L_{\text{moe}}N/C$ experts and broadcasts to $C-1$ peers, the total broadcast volume per full refresh wave is

$$\frac{L_{\text{moe}}N}{C}(C-1) \cdot 2q(d_e+r) \cdot \text{bytes}.$$

This traffic is small relative to token-level expert all-to-all and is not on the per-layer critical path.

13.5 Taylor Proxy as a Trust-Region Model

The worker objective can be viewed as a trust-region approximation. Let $z = E_e(x; \theta_e)$ be the expert output and $g_z = \partial \mathcal{L} / \partial z$ be the captured output gradient. For a candidate expert update $\Delta\theta$, the first-order change in loss is approximated by

$$m(\Delta\theta) = \langle g_z, E_e(x; \theta_e + \Delta\theta) - E_e(x; \theta_e) \rangle + \frac{\lambda}{2} \|\Delta\theta\|_2^2.$$

The first term keeps the exact nonlinear expert forward under the candidate weights, while linearizing only the downstream loss. The second term limits step size and absorbs curvature/model-mismatch risk. The unproven but testable condition is model adequacy: within the accepted update radius, proxy improvement should correlate with true training improvement often enough to be useful.

13.6 Error Feedback Intuition

When expert deltas are sparsified or quantized, error feedback keeps the residual from discarded coordinates and adds it to the next candidate update. For top-α sparsification, the residual norm is bounded by the energy outside the largest coordinates:

$$\|x - \mathrm{TopK}_\alpha(x)\|_2^2 \leq (1 - \alpha)\|x\|_2^2.$$

This does not prove convergence of the full asynchronous worker system. It only gives the intuition for retaining residual information across repeated sparse expert updates.

14 Appendix: Systems Requirements

14.1 Composer Requirements

A composer must hold the always-active model state, owned expert shards, non-owned surrogates, optimizer state for trainable on-composer parameters, and enough activation memory for local training. In the direct variant, it also holds optimizer state for owned routed experts. Under offload, routed-expert optimizer state can be absent from the composer. This is the main memory advantage: expert Adam state scales as $1/C$ in the direct variant and can be zero on the composer when experts are offloaded.

14.2 Worker Requirements

A worker needs only the assigned expert, the local optimizer state for that expert, one or more sketches, and temporary buffers. It does not need a tokenizer, a data shard, the full model, the router, the backbone, or other experts. New workers can be assigned expert-local work after receiving a small state bundle rather than a full model checkpoint.

14.3 Network Requirements

Parallax replaces a latency-sensitive expert all-to-all with several bandwidth-tolerant background channels:

- tiered non-expert synchronization through the syncer;
- surrogate refresh from owners to composers;
- sketch dispatch from composers to workers;
- compressed expert updates from workers to owners or composers;
- occasional full refreshes for version correction or worker bootstrap.

The system should be engineered around durable queues and versioned, retryable, idempotent updates rather than around low-jitter collectives.

14.4 Failure Handling

Composer failure and worker failure have different consequences. A worker failure drops or delays expert-local work; the composer can continue training with existing expert and surrogate state. A composer failure is more serious because it owns expert shards and participates in tiered synchronization. The hard-ownership implementation should checkpoint owned experts, surrogate versions, and outer-sync state frequently enough that a replacement composer can resume ownership. Replicated ownership, discussed in future work, would reduce this risk further by maintaining multiple live owners per expert shard.

14.5 Security and Permissionless Extensions

The present paper uses managed workers, and the decomposition is compatible with permissionless workers as a future direction. Workers see sketches rather than raw data and full models. Trust can be built from version checks, proxy-improvement reports, update norm statistics, sampled redundant work, and held-out audit sketches. This is a systems boundary for future auditing work; the paper does not claim a formal privacy or Byzantine fault-tolerance guarantee.

15 Appendix: Large-Model Architecture Plan

15.1 Design Objective

The large-model plan is not simply "scale the 20B run." The objective is to combine the PARALLAX training decomposition with an architecture that is economical at inference time. A model with 176B total parameters and roughly 8.7B active parameters has a different serving profile from a dense 176B Transformer. If most capacity lives in sparse ternary experts and much of the backbone uses Mamba-style state-space layers, the model can offer high memorized and specialized capacity while keeping active compute, KV-cache footprint, and decode latency under control.

15.2 176B Target Configuration

Table 14: Planned 176B architecture used for feasibility and scaling analysis. This is a design target rather than a completed convergence result.

Component	Target value
Total parameters	$\sim$176B
Active parameters/token	$\sim$8.7B
MoE layers	32
Routed experts/layer	640
Experts/token	16
Expert hidden shape	$1024 \to 4096 \to 1024$
LatentMoE dimension	1024
Backbone pattern	Mamba-3 MIMO heavy, selective attention
Residual transport	Manifold-constrained hyper-connections
Expert format	Ternary / low-bit serving target
Composer count tested for smoke run	4

The design deliberately scales expert width rather than only expert count. Wider experts improve arithmetic intensity and make each expert a more capable specialist. Keeping top-$k = 16$ preserves a fixed active-expert count while total capacity grows. LatentMoE keeps the expert interface dimension at 1024, so sketch traffic, surrogate storage, and all-to-all accounting remain bounded relative to full model width.

15.3 Mamba-3 MIMO Role

Mamba-3 MIMO is included for inference efficiency. Attention layers require KV-cache storage and attention computation that grow with context. State-space layers maintain recurrent state and can decode with a much flatter dependence on context length. A Mamba-heavy architecture attacks the serving half of the problem directly: the model should be cheap to train and able to run long-context agents without a prohibitive KV-cache bill.

MIMO improves channel mixing and raises the capacity of the state-space path. The planned architecture does not remove attention entirely because attention remains useful for exact retrieval and for some long-range binding patterns. The intended ratio is hybrid: enough attention to close quality gaps, enough Mamba to make long-context inference efficient.

15.4 Manifold-Constrained Hyper-Connections

Deep sparse/hybrid models need stable residual dynamics. Manifold-constrained hyper-connections replace a simple residual stream with constrained mixing across multiple streams. The constraint keeps residual transport bounded while allowing richer inter-stream routing of information. In the future PARALLAX architecture, expert ownership

and surrogate recall solve the distributed training problem; stable residual transport solves a separate depth/stability problem.

15.5 Ternary Experts as Serving Infrastructure

Ternary routed experts are included because expert memory is the dominant total-parameter storage cost. If expert weights can be packed into a tiny codebook with per-tensor or per-channel scales, serving can keep many more experts resident or move them more cheaply. The expert interface remains dense enough for tensor-core-friendly execution after dequantization or direct low-bit kernels.

16 Appendix: Experimental Roadmap

16.1 Composer Count Sweep

The immediate controlled sweep should test $C = 4, 8, 16, 32$ while holding the number of expert shards fixed. This separates two effects that are otherwise conflated: more composers and smaller owned expert fractions. Replicated ownership is the key design to test. For example, 32 composers can map onto 8 expert shards, creating four replicas per shard. The experiment should compare hard single-owner shards, two owners per shard with simple averaging, four owners per shard with simple averaging, co-owner RDA-DiLoCo with outer momentum, and co-owner averaging with surrogate-consistency penalties.

Replication increases memory and compute per shard, but reduces owner noise, improves fault tolerance, and provides more reliable surrogate refreshes. The correct setting likely depends on composer count and data heterogeneity.

16.2 Surrogate Rank and Refresh Sweep

Surrogate rank should be swept jointly with refresh cadence. The useful metric is not reconstruction error alone. The right metric is validation loss at fixed composer compute and fixed background bandwidth. A practical sweep should vary $r \in \{16, 32, 64, 96, 128\}$ and refresh horizons $\{5, 10, 20, 50\}$. For each point, report surrogate forward FLOPs, surrogate storage per composer, surrogate update bandwidth, validation-loss gap, router entropy, expert load balance, and sensitivity to delayed refresh.

The likely optimum is not the highest rank. A slightly biased but cheap surrogate can be better than an accurate surrogate that consumes too much compute or refresh bandwidth.

16.3 Worker Offload Sweep

Worker offload needs its own matrix of experiments. The variables are sketch compression, number of sketches per update, inner optimizer, update sparsity, version lag, and acceptance threshold. The sweep should report both model quality and systems metrics: worker updates accepted per hour, accepted update norm distribution, proxy improvement before and after compression, stale-update rejection rate, composer time spent on sketch capture, worker GPU memory and utilization, and bandwidth per accepted expert update.

In the mature system, a failed worker should reduce expert update throughput without interrupting composer training. That property should be measured directly by injecting worker churn.

16.4 Inference Evaluation

Every large Parallax model should be evaluated as an inference system. Required serving metrics include validation loss, prefill tokens/sec at several context lengths, decode tokens/sec at realistic batch sizes, KV-cache memory per concurrent sequence, active expert memory bandwidth per generated token, expert cache behavior, cost per million generated tokens, quality as a function of Mamba/attention ratio, and long-context agentic workflow benchmarks.

Hybrid Mamba-attention MoE systems are the relevant serving family for future comparisons: quality, long context, and throughput need to be measured together. Parallax should be judged by whether it can train models in that family using fragmented compute.

17 Appendix: Full-System Comparison

17.1 Ordinary Expert Parallelism

Ordinary expert parallelism is the right baseline for colocated clusters. It keeps expert computation exact and lets the router select any expert. Its weakness is physical: token activations must travel to expert owners every MoE layer. On a fast intra-datacenter fabric this can be engineered. Across wide-area links it becomes the wrong abstraction. The training step is held hostage by the slowest remote expert traffic.

17.2 Full-Model Decentralized Training

Full-model decentralized training reduces the frequency or precision of model synchronization, but each participant still trains a full model replica or a large shard. This is viable for dense models and some colocated sparse settings. It is less compelling for sparse MoE pretraining because routed experts dominate total parameters while only a small subset is active per token. Treating all parameters as one synchronization object misses the opportunity created by expert sparsity.

17.3 Parallax Decomposition

Parallax changes the unit of work:

- composers train a locally complete sparse model with owned experts and surrogate recall;
- the syncer coordinates non-expert state by tier;
- owners authoritatively refresh surrogates for their expert shards;
- workers optimize expert-local objectives from sketches.

Training is efficient because expensive expert state is sharded, approximated locally where it is not owned, and offloadable to expert workers. The serving implications of sparse/low-bit experts are a useful downstream motivation, but the contribution evaluated in this paper is the training decomposition.

17.4 Combined Decomposition

Each component solves a different training bottleneck. Owned experts reduce memory and optimizer state. Surrogates preserve global routing while eliminating remote dispatch. Tiered synchronization keeps global interfaces aligned. Sketch workers remove expert update work from composers. LatentMoE compresses the expert interface. Ternary experts reduce stored capacity. Removing any one component leaves a gap:

- without surrogates, composers are not locally complete;
- without ownership, expert state does not scale down;
- without tiered sync, routers and latent projections drift;
- without workers, composers still pay expert optimizer cost;
- without LatentMoE, sketches and surrogates become too wide;
- without low-bit experts, expert storage and worker/composer memory pressure rise.

18 Conclusion

Parallax decomposes sparse MoE training around expert ownership. A composer trains against the shared expert namespace while executing only owned experts and low-rank surrogates. Non-owned expert backward passes disappear from the composer. Synchronous expert all-to-all leaves the critical path. Routed-expert optimizer state can move to workers. The participant becomes smaller as the hierarchy gets wider.

The completed 20B runs are the main evidence in this report. The H100 run reaches the B300 baseline point estimate at the matched 30k-equivalent token count. The worker-offload run is noisier, but it keeps improving while remote workers carry routed-expert update work. The result is a bounded one: coherent 20B sparse training works in the tested decentralized settings. The cost model explains why that matters: more composers can shrink local expert compute, expert memory, and optimizer state while also avoiding the remote-token traffic imposed by standard expert-parallel training. Larger composer counts, replicated ownership, optimized worker throughput, and long 176B training runs remain open engineering and experimental work.

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.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, and Torsten Hoefler. HIGGS: Homogeneous and isotropic gaussian grouping for quantization. *arXiv preprint arXiv:2411.17525*, 2024a.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. QuaRot: Outlier-free 4-bit inference in rotated LLMs. *arXiv preprint arXiv:2404.00456*, 2024b.

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Anima Anandkumar. signSGD: Compressed optimisation for non-convex problems. *International Conference on Machine Learning*, 2018.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. QuIP: 2-bit quantization of large language models with guarantees. *Advances in Neural Information Processing Systems*, 36, 2024.

Aidan Clark, Diego de las Casas, Aurelia Guy, Arthur Sherwood, Thomas L. Griffiths, Jacob Menick, et al. Unified scaling laws for routed language models. *arXiv preprint arXiv:2202.01169*, 2022.

Damai Dai, Chengqi Deng, Chenggang Zhao, R.X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, et al. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. *International Conference on Machine Learning*, 2024.

DeepSeek-AI. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Michael Diskin, Alexei Bukhtiyarov, Max Ryabinin, Lucile Saulnier, Quentin Siber, Nils Thakur, Ben Verhoeven, and Thomas Wolf. Distributed deep learning in open collaborations. *Advances in Neural Information Processing Systems*, 34, 2021.

Arthur Douillard, Qixuan Feng, Andrei A Ruber, Raviteja Vemulapalli, Hakan Bilen, and Nicolas Carion. DiLoCo: Distributed low-communication training of language models. *arXiv preprint arXiv:2311.08105*, 2023.

Arthur Douillard, Yanislav Donchev, Keith Rush, Satyen Kale, Zachary Charles, Zachary Garrett, Gabriel Teston, Dave Lacey, Ross McIlroy, Jiajun Shen, Alexandre Ramé, Arthur Szlam, Marc'Aurelio Ranzato, and Paul Barham. Streaming DiLoCo with overlapping communication. *arXiv preprint arXiv:2501.18512*, 2025a.

Arthur Douillard et al. Decoupled DiLoCo: Scaling distributed optimization with component-level communication. *arXiv preprint arXiv:2604.21428*, 2025b.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, et al. GLaM: Efficient scaling of language models with mixture-of-experts. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pages 5547–5569. PMLR, 2022.

Venmugil Elango, Nidhi Bhatia, Roger Waleffe, Rasoul Shafipour, Tomer Asida, Abhinav Khattar, Nave Assaf, Maximilian Golub, Joey Guman, Tiyasa Mitra, et al. LatentMoE: Toward optimal accuracy per FLOP and parameter in mixture of experts. *arXiv preprint arXiv:2601.18089*, 2026.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. *International Conference on Learning Representations*, 2023.

Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. MegaBlocks: Efficient sparse training with mixture-of-experts. *Proceedings of Machine Learning and Systems*, 5, 2023.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Jiaao He, Jiezhong Zhai, Tiago Antunes, Haojun Wang, Fubing Luo, Shangfeng Shi, and Qin Li. FastMoE: A fast mixture-of-expert training system. *arXiv preprint arXiv:2103.13262*, 2021.

Qirong Ho, James Cipar, Henggang Cui, Seunghak Lee, Jin Kyu Kim, Phillip B Gibbons, Garth A Gibson, Greg Ganger, and Eric P Xing. More effective distributed ML via a stale synchronous parallel parameter server. *Advances in Neural Information Processing Systems*, 26, 2013.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. *International Conference on Learning Representations*, 2022.

Changho Hwang, Wei Cui, Yifan Xiong, Ziyue Yang, Ze Liu, Han Hu, Zilong Wang, Rafael Saez, Olatunji Ruwase Alon, and Minjia Zhang. Tutel: Adaptive mixture-of-experts at scale. In *Proceedings of Machine Learning and Systems*, volume 5, 2023.

Max Jaderberg, Wojciech Marian Czarnecki, Simon Osindero, Oriol Vinyals, Alex Graves, David Silver, and Koray Kavukcuoglu. Decoupled neural interfaces using synthetic gradients. In *International Conference on Machine Learning*, 2017.

Arthur Jaghouar, Dmitry Baranchuk, et al. OpenDiLoCo: An open-source framework for globally distributed low-communication training. *arXiv preprint arXiv:2407.07852*, 2024.

Sami Jaghouar, Justus Mattern, Jack Min Ong, Jannik Straube, Manveer Basra, Aaron Pazdera, Kushal Thaman, Matthew Di Ferrante, Felix Gabriel, Fares Obeid, et al. INTELLECT-2: A reasoning model trained through globally decentralized reinforcement learning. *arXiv preprint arXiv:2505.07291*, 2025.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error feedback fixes SignSGD and other gradient compression schemes. *International Conference on Machine Learning*, 2019.

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

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

Jakub Krajewski, Jan Ludziejewski, Kamil Adamczewski, Maciej Piotrowski, Michał Krutul, Sławomir Antoniak, Kamil Ciebiera, et al. Scaling laws for fine-grained mixture of experts. *arXiv preprint arXiv:2402.07871*, 2024.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. GShard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2021.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. BASE layers: Simplifying training of large, sparse models. In *International Conference on Machine Learning*, 2021.

Fengfu Li, Bo Zhang, and Bin Liu. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.

Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. *USENIX Symposium on Operating Systems Design and Implementation*, 2014.

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, 2020.

Joel Lidin, Amir Sarfi, Erfan Miahi, Quentin Anthony, Shivam Chauhan, Evangelos Pappas, Benjamin Thérien, Eugene Belilovsky, and Samuel Dare. Covenant-72b: Pre-training a 72b LLM with trustless peers over-the-internet. *arXiv preprint arXiv:2603.08163*, 2026.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for LLM compression and acceleration. *Proceedings of Machine Learning and Systems*, 6, 2024.

Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. *International Conference on Learning Representations*, 2018.

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

Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. The era of 1-bit LLMs: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764*, 2024.

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

Konstantin Mishchenko, Francis Bach, Mathieu Even, and Blake Woodworth. Asynchronous SGD revisited: Sharper rates and optimal delay. *arXiv preprint arXiv:2206.07638*, 2022.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, et al. OLMoE: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*, 2024.

Nous Research. Psyche: Decentralized AI training on solana. https://psyche.network/, 2025. Production deployment of DisTrO for decentralized pretraining.

NVIDIA. NVIDIA nemotron 3: Efficient and open intelligence. *arXiv preprint arXiv:2512.20856*, 2025.

Bowen Peng. A preliminary report on DisTrO. https://github.com/NousResearch/DisTrO/blob/main/A_Preliminary_Report_on_DisTrO.pdf, 2024. Nous Research preliminary report.

Prime Intellect. INTELLECT-1: Launching the first decentralized training of a 10b parameter model. *arXiv preprint arXiv:2410.06003*, 2024.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. ZeRO: Memory optimizations toward training trillion parameter models. *International Conference on High Performance Computing, Networking, Storage and Analysis*, 2020.

Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. DeepSpeed-MoE: Advancing mixture-of-experts inference and training to power next-generation AI scale. In *International Conference on Machine Learning*, 2022.

Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. *Advances in Neural Information Processing Systems*, 24, 2011.

Peter Richtárik, Igor Sokolov, and Ilyas Fatkhullin. EF21: A new, simpler, theoretically better, and practically faster error feedback. *Advances in Neural Information Processing Systems*, 34, 2021.

Max Ryabinin and Anton Gusev. Towards crowdsourced training of large neural networks using decentralized mixture-of-experts. In *Advances in Neural Information Processing Systems*, volume 33, pages 3659–3672, 2020.

Mika Senghaas, Fares Obeid, Sami Jaghouar, William Brown, Jack Min Ong, Daniel Auras, Matej Sirovatka, Jannik Straube, Andrew Baker, Sebastian Mueller, et al. INTELLECT-3: Technical report. *arXiv preprint arXiv:2512.16144*, 2025.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Mahdavi, Andrew Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017.

Sebastian U Stich. Local SGD converges fast and communicates little. *arXiv preprint arXiv:1805.09767*, 2019.

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

Hanlin Tang, Shaoduo Gan, Samyam Rajbhandari, Xiangru Li, and Yuxiong He. 1-bit Adam: Communication efficient large-scale training with Adam's convergence speed. *International Conference on Machine Learning*, 2021.

Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. PowerSGD: Practical low-rank gradient compression for distributed optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Huaijie Wang, Lingxiao Ma, Fan Yang, Ruiping Wang, Yi Wu, and Furu Wei. BitNet: Scaling 1-bit transformers for large language models. *arXiv preprint arXiv:2310.11453*, 2023.

Zhaohui Wang et al. SparseLoCo: Sparse low-communication training for decentralized LLM pre-training. *NeurIPS Workshop*, 2024.

Leyang Xue, Yao Jia, Zhi Zhang, Zhen Wen, and Wenqiang Shan. MoE-Infinity: Offloading-efficient MoE model serving. *arXiv preprint arXiv:2401.14361*, 2024.

Amir Zandieh, Majid Daliri, Majid Hadian, and Vahab Mirrokni. TurboQuant: Online vector quantization with near-optimal distortion rate. *arXiv preprint arXiv:2504.19874*, 2025.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. GaLore: Memory-efficient LLM training by gradient low-rank projection. *International Conference on Machine Learning*, 2024.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, et al. PyTorch FSDP: Experiences on scaling fully sharded data parallel. *Proceedings of the VLDB Endowment*, 16(12), 2023.

Shuxin Zheng, Qi Meng, Taifeng Wang, Wei Chen, Nenghai Yu, Zhi-Ming Ma, and Tie-Yan Liu. Asynchronous stochastic gradient descent with delay compensation. *International Conference on Machine Learning*, 2017.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, and James Laudon. Mixture-of-experts with expert choice routing. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. In *International Conference on Learning Representations*, 2017.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. ST-MoE: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.