REPS: COMPRESSED EXPERT REPLACEMENT FOR MULTIVERSION MOE SERVING

Anonymous authors
Paper under double-blind review

ABSTRACT

Serving expert-local customizations of a mixture-of-experts (MoE) model requires a representation that supports both local updates and shared execution. We propose Replaceable Expert Patches for Serving (REPS), which uses the same compressed records for a parent and its complete expert replacements. Two sign bytes encode each width-eight block through a complete Hadamard transform. This structure gives independent scalar assignment at fixed scale and integer reconstruction without codeword lookup. A version-aware operator selects parent or patch rows and consumes these records directly during prefill. Qwen3-30B-A3B Code updates improve target perplexity while keeping non-target and general-text degradation within 1% in three fixed-initialization data orders. On one of these exact updates, integer reconstruction reduces synchronized backend first-token time by 3.51× and 4.45× at batches one and eight relative to our lookup-table (LUT) decoder, with unchanged weights and decode. In a nine-format storage control, sharing reduces REPS weight allocation by 48.9% and 73.4% at two and four structural versions.

1 INTRODUCTION

Deploying mixture-of-experts (MoE) models for diverse tasks often requires serving multiple customized versions of a common backbone. Deploying each version independently duplicates unchanged weights in GPU memory, making memory consumption grow unnecessarily with the number of versions. Sharing unchanged weights offers a promising way to reduce this overhead (Sheng et al., 2024; Chen et al., 2024; Wu et al., 2024). For MoE models whose customizations modify only selected experts, unchanged experts can be stored once and reused across versions. Shared serving must distinguish the expert weights used by different versions. The same expert index may correspond to different weights in different versions. How can these expert weights be stored compactly while enabling efficient joint execution of multiple model versions?

Shared execution must combine request version with token-level expert selection. A request retains its version throughout generation, while the router chooses experts for each token. Two requests routed to expert 3 can therefore require different weights. The operator must select the correct parent or patch rows.

Expert-local fine-tuning provides the update pattern studied here. It changes selected experts while freezing shared parameters (Wang et al., 2024). Qwen3-30B-A3B has 128 experts per layer (Qwen Team, 2025). Updating four leaves 124 experts, the router and attention reusable.

Prior systems already serve related versions over shared weights. SLoRA, Punica and dLoRA support low-rank adaptation (LoRA) (Hu et al., 2022; Sheng et al., 2024; Chen et al., 2024; Wu et al., 2024). ExpertWeave supports complete expert replacements (Shi et al., 2025). Quantization reduces weight storage (Frantar et al., 2023; Lin et al., 2024; Egiazarian et al., 2024; Tseng et al., 2024a). We study how complete expert replacements can be encoded independently and consumed by a common compressed operator.

REPS derives encoding and prefill reconstruction from one compressed block family. It applies a complete width-eight Hadamard transform to a four-level Cartesian grid, storing each grid point in two sign bytes. The complete grid makes nearest assignment separable at fixed scale. The same bytes admit an exclusive-OR and population-count reconstruction rule. Assignment and prefill thus use the same discrete coordinates, without searching or loading a codeword table.

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

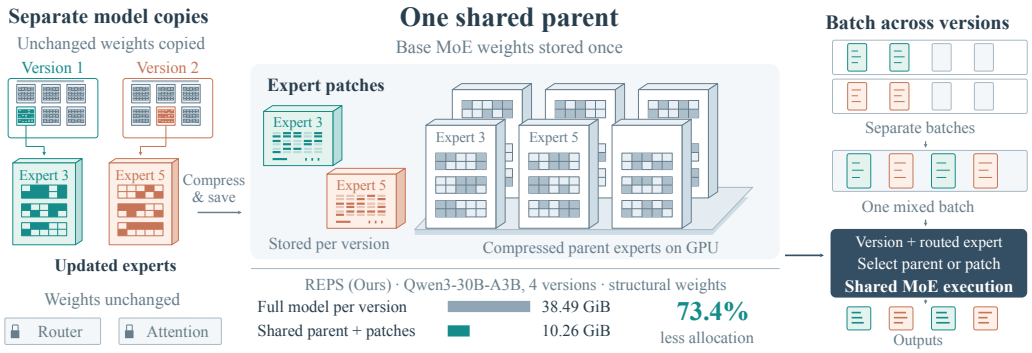


Figure 1: REPS (Ours) stores selected expert replacements while versions share one resident parent. Request version and token routing select parent or patch rows for the same packed operator. Allocation uses four structural versions.

REPS keeps expert replacements local through independent, fixed-stride records. Each projection stores its own codes and scale. An expert replacement changes these records while retaining the decoder and layout. Request version and token routing select parent or patch rows for the same packed operator, as illustrated in Figure 1. The encoding rule therefore produces records that the resident service can consume directly.

We introduce REPS, a compressed expert representation whose coordinates determine both independent assignment and packed reconstruction. We develop the representation, replacement records and serving operator as follows.

- We derive a **sign-byte representation with separable assignment and integer reconstruction**. Its complete width-eight grid reduces fixed-scale nearest assignment to a small transform and scalar decisions. The same bytes reconstruct matrix tiles by bit operations, giving local encoding without codebook fitting and prefill without codeword lookup.
- We propose **independently replaceable expert records** in this representation. Per-projection codes and scales preserve the unchanged parent. Fixed-stride records let the runtime address parent or patch rows directly, so each added version stores its changed experts without a separate model copy.
- We develop a **version-aware packed prefill operator** that selects records jointly by request version and routed expert, then reconstructs them inside the matrix multiplication. Parent and patch rows use the same operator, so different versions can share a batch. A same-weight decoder intervention isolates integer reconstruction's reduction in backend first-token time.

Experiments on Qwen3-30B-A3B and Trinity-26B-A3B measure encoding, weight residency and execution, alongside deployment quality and latency.

2 Related Work

Low-bit weight representations. Low-bit representations trade assignment cost against reconstruction flexibility. Additive Quantization for Language Models (AQLM) fits additive codebooks and assigns codes across transformer blocks for flexible low-bit reconstruction (Egiazarian et al., 2024). QTIP increases the effective quantization dimension with a stateful trellis, while LLVQ uses the 24-dimensional Leech lattice (Tseng et al., 2024b; van der Ouderaa et al., 2026). Both avoid large dense codebooks but require trellis or lattice search for assignment. HIGGS combines Hadamard rotations with fixed grids without calibration data and can move the transform to activations for computation in the rotated space (Malinovskii et al., 2025). Qift studies the same four scalar levels used here for rotated two-bit weights (Huang & Tsai, 2026). REPS uses a complete width-eight grid whose sign bytes support separable fixed-scale assignment and integer tile reconstruction, connecting local encoding to prefill through the same records.

Compression of MoE experts. MoE compression targets the storage that remains despite sparse activation (Rajbhandari et al., 2022; Frantar & Alistarh, 2024). Sharing structure across experts can

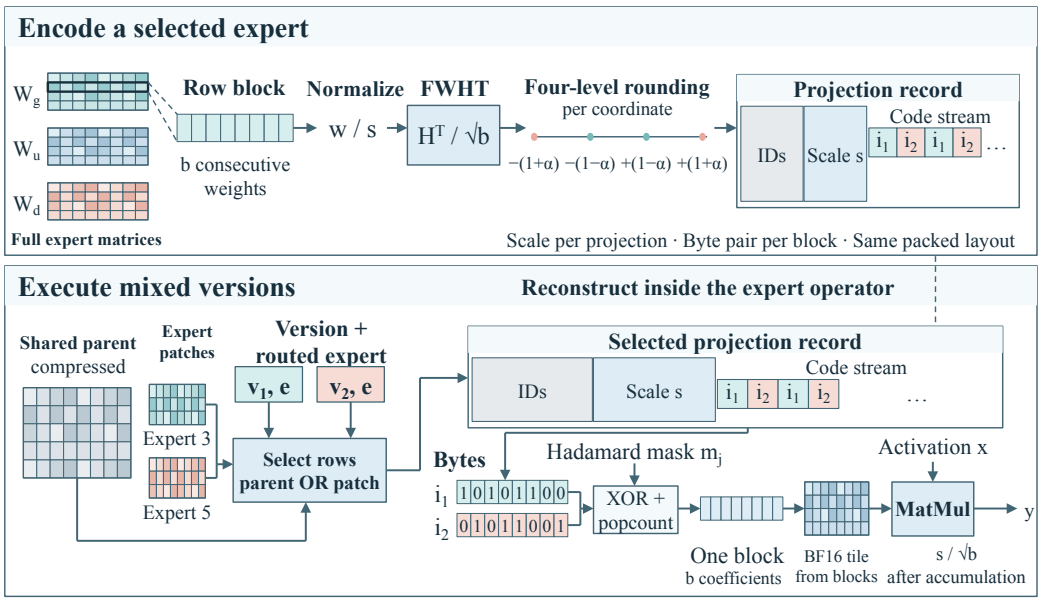


Figure 2: REPS (Ours) connects expert encoding to compressed execution. Each projection stores one scale and a stream of sign-byte pairs, one pair per block. Parent and replacement records share this format. Version and expert selection choose the record consumed by integer prefill reconstruction.

reduce this storage. D^2-MoE uses a shared base and low-rank residuals without additional training (Gu et al., 2025). BitsMoE shares a spectral basis and assigns precision to expert-specific factors under a bit budget (Zhao et al., 2026). Changing these shared components can affect several experts. REPS instead stores full-dimensional projections independently, giving up shared-basis compression to preserve unselected slices during replacement. QMoE combines extreme compression with custom decoding kernels (Frantar & Alistarh, 2024).

Efficient adaptation. Efficient adaptation limits which weights change and how updates are stored. Low-rank adaptation (LoRA) trains small factor matrices over a frozen parent (Hu et al., 2022). QLoRA reduces training memory with a frozen four-bit parent (Dettmers et al., 2023). These compact updates are constrained by rank and placement. Expert-specialized fine-tuning selects task-relevant MoE experts and freezes other modules (Wang et al., 2024). REPS exports these changed experts in the served format. Its patches can exceed LoRA adapters in size but use the expert operator without a separate adapter branch.

Serving related model versions. Related-version serving shares unchanged weights while dispatching requests to their selected customization. This weight reuse complements paged key–value caches (Kwon et al., 2023). SLoRA pages adapters with the key–value cache, Punica batches LoRA computation across tenants, and dLoRA coordinates requests and adapters (Sheng et al., 2024; Chen et al., 2024; Wu et al., 2024). DeltaZip compresses full-model fine-tuning deltas. BitDelta batches one-bit deltas over a shared base (Yao et al., 2025; Liu et al., 2024). ExpertWeave shares unchanged weights and batches complete expert replacements. Its rerouting maps each token's version and routed expert into a virtual expert tensor. Virtual-memory management allocates storage for active expert regions (Shi et al., 2025). Within this setting, REPS gives parent and patch rows a common fixed-stride layout for independent encoding and integer prefill reconstruction. Frozen-book AQLM supports the same replacement interface. Mixing efficiency depends on arrival patterns and executed batch sizes.

3 METHODOLOGY

REPS encodes independent expert replacements for a common compressed operator (Figure 2).

3.1 EXPERT-LOCAL VERSION REPRESENTATION

Expert-local versions share unchanged weights and replace complete projections. Let B_0 denote the compressed parent and $\mathcal{P}_v$ the replacements for version v's changed layer–expert pairs $\mathcal{U}_v$. Records store these identifiers (IDs), unsigned eight-bit (uint8) indices I and a bfloat16 (BF16) scale s for each gate, up or down projection $p \in \{g, u, d\}$. Execution selects

$$\widehat{B}^p_{v,\ell,e} = \begin{cases} \mathcal{P}_v[\ell, e, p], & (\ell, e) \in \mathcal{U}_v, \\ B_0[\ell, e, p], & \text{otherwise.} \end{cases} \tag{1}$$

Expert-local adaptation freezes the router, attention and other shared parameters. Changing them requires a new parent.

Independent encoding preserves unselected bytes within one encoder build and numerical path. Expert scales and indices are computed separately. Only immutable format constants are shared.

3.2 JOINT VERSION SELECTION AND TOKEN ROUTING

Using the common record layout, joint version and expert selection lets one compressed operator batch different versions of the same expert. The request retains version v. At layer ℓ, state x routes to experts $\mathcal{S}$ with weights a_e. Equation 1 selects their records. With version and layer suppressed, let U^p_e and s^p_e denote the unscaled projection and its scale. The expert operation is

$$y = \sum_{e \in \mathcal{S}} a_e s^d_e U^d_e \big[\mathrm{SiLU}(s^g_e U^g_e x) \odot (s^u_e U^u_e x)\big], \tag{2}$$

where SiLU is the sigmoid linear unit activation, with the gate scale inside it.

3.3 FIXED-STRIDE LAYOUT AND EXPERT ENCODING

Fixed-size blocks let the runtime locate any parent or patch row directly. Each projection has the form $W^p_e = s^p_e Q_\nu(I^p_e)$, where ν specifies the fixed decoder format. Q_ν assembles a projection from the block decoder q_ν below. Let b be the block width and α the relative weight of the second sign plane. We use $b = 8$ and $\alpha = 1/2$. Each byte i encodes a sign vector $c(i) \in \{-1, +1\}^b$, with set bits denoting positive signs. The Hadamard matrix satisfies $H_\nu^\top H_\nu = bI$, with identity I. The block for bytes i_1, i_2 is

$$q_\nu(i_1, i_2) = \frac{1}{\sqrt{b}} H_\nu (c(i_1) + \alpha c(i_2)) \,. \tag{3}$$

Orthogonal normalization by $1/\sqrt{b}$ preserves distances. $\alpha = 1/2$ gives the coordinate grid $\{-3/2, -1/2, 1/2, 3/2\}$. This block family yields assignment (Eq. 4) and integer reconstruction (Eq. 5).

Independent projection encoding needs no learned codebook, rotation seed, Hessian or calibration set. Projection scales add less than 1.1×10^{-5} bits per expert parameter (bpp), yielding 2.00 expert bpp. Non-expert BF16 tensors are counted separately. Input widths must be divisible by eight.

Encoding and adaptation. Encoding assigns each block independently at its fixed projection scale. Let $\mathcal{I}_\nu$ contain the legal projection index arrays, with one byte pair per block. Using the Euclidean norm on flattened entries, the assignment is

$$I^p_e = \mathcal{A}_\nu(W^p_e / s^p_e), \qquad \mathcal{A}_\nu(x) = \arg\min_{i \in \mathcal{I}_\nu} \|x - Q_\nu(i)\|_2^2. \tag{4}$$

Orthogonality reduces nearest-grid assignment to scalar rounding after a fast Walsh–Hadamard transform (FWHT), with zero and midpoint ties choosing the lower adjacent value. For D entries and $b = 8$, it costs $O(D \log b)$ transform arithmetic and $O(D)$ scalar decisions without a codeword table. Exact nearest assignment assumes a fixed positive scale and exact arithmetic. Scale selection and finite-precision casts are separate. Near-rate AQLM searches two frozen 256-entry learned books.

For initial encoding, we match the spread of a projection's weights to that of the first sign plane. Let σ_1 be sample standard deviation over all scalar entries and let $P_0 = \{H_\nu c/\sqrt{b} : c \in \{-1, 1\}^b\}$. We initialize the scale as $s_0 = \sigma_1(W)/\sigma_1(P_0)$. The 2^b-vector pool has standard deviation near one, so

the initializer tracks weight spread while leaving full-grid error unoptimized. Appendix S1 specifies numerical safeguards, endpoint-specific scale casts and device reductions.

Parent continuation then addresses the substantial quality loss from direct assignment. The served Qwen3-30B-A3B parent retains fixed expert codes while updating shared weights and, in later stages, per-matrix gains.

Expert-local adaptation fixes parent scales and shared weights. Selected-expert quantization-aware training (QAT) updates 32-bit floating-point (FP32) weight-domain latents with straight-through gradients, refreshing assignments once per optimizer step. The dense training path exports uint8 indices and retained BF16 scales for packed execution.

Reconstruction inside the operator. Integer reconstruction lets prefill consume sign bytes inside matrix multiplication. For the Sylvester matrix H_ν, let m_j mark the positive entries of row j. The block is reconstructed as

$$[q_\nu(i_1, i_2)]_j = \frac{b(1+\alpha) - 2\,\mathrm{popc}(i_1 \oplus m_j) - 2\alpha\,\mathrm{popc}(i_2 \oplus m_j)}{\sqrt{b}}, \tag{5}$$

where popc counts set bits and $\oplus$ denotes exclusive OR (XOR). A length-b sign-vector dot product with d mismatches is $b - 2d$. Weighting the two dot products by 1 and α gives the numerator.

Prefill reconstructs sign-byte tiles with integer coefficients, casts them to BF16 for multiplication, and applies normalization with the projection scale after accumulation. Fixed arithmetic replaces codeword-table accesses while retaining packed records. Per-MoE CUDA Graphs reduce launch overhead.

The same-payload comparison changes only prefill decoding, preserving weights, scales, tile shape, routing and autoregressive decode.

4 EXPERIMENTS

We evaluate compressed execution, weight sharing and adaptation quality, testing trained versions for target improvement and retention before interpreting serving results.

Setup. Three MoE backbones supply training and deployment controls. OLMoE-1B-7B-0924 (Muennighoff et al., 2025) (64 experts, top-8) supplies matched reference-layout training and a no-reference pair with different devices and world sizes. Qwen3-30B-A3B-Instruct-2507 (Qwen Team, 2025) (128 experts, top-8) supplies deployment measurements. Trinity-26B-A3B, the Trinity-Mini-Base checkpoint (Arcee AI, 2025), is the second deployment backbone. For the deployment backbones, names give total then active parameters per token; A3B means approximately three billion active parameters. Only expert feed-forward networks (FFNs) are quantized.

Perplexity (PPL) measures quality on non-overlapping length-2048 WikiText-2 (Merity et al., 2017) and Colossal Clean Crawled Corpus (C4) (Raffel et al., 2020) windows. The four-corpus evaluation adds Penn Treebank (PTB) and PubMed. Repeated measurements report sample standard deviation (SD).

Endpoints. Studies distinguish served, adapted and codec-control endpoints. Table 1 serves the 10.33-GB Qwen3-30B-A3B payload without a reference; REPS quality averages three final-stage data orders sharing an earlier checkpoint. The paired Qwen3-30B-A3B study starts from the served parent; the earlier locality control uses a separate latent parent. Codec controls rebuild from a common step-250 checkpoint. Appendix S2 and S6 give training and evaluation protocols.

4.1 SINGLE-MODEL QUALITY AND EXECUTION

Table 1 compares served Qwen3-30B-A3B implementations at batch eight by PPL, output rate and time to first token (TTFT). Precision, weights, training history and execution differ. BF16/NF4 use eager execution. HIGGS/LLVQ/KBVQ reconstruct to BF16. QTIP/QuIP# use adapted author kernels. AQLM/REPS use captured Tensor Core paths. The same-weight control below isolates the decoder contribution.

Table 1: Qwen3-30B-A3B implementations. Serving uses an A100-80GB at batch size 8, with 128 input tokens and 32 generated tokens per request. Quality is perplexity. Rate is batch output tokens/s. TTFT is time to first token in ms. Quality SD covers sampled training or quantization settings and is inapplicable to fixed endpoints. Serving repeats are defined in the text.

Method	bpp ↓	Wiki-2 ↓	C4 ↓	PTB ↓	PubMed ↓	Rate ↑	TTFT ↓
BF16	16.000	7.26	**12.56**	12.43	5.78	14.6 $\pm$0.02	1463 $\pm$3
NF4 (Dettmers et al., 2023)	4.127	7.36	12.75	12.55	5.83	6.2 $\pm$0.02	3774 $\pm$5
LLVQ (van der Ouderaa et al., 2026)	5.183	8.47	14.40	14.37	6.35	1.9 $\pm<$ 0.01	11477 $\pm$52
KBVQ (Xu et al., 2026)	2.799	8.54 $\pm$0.04	14.31 $\pm$0.03	14.30 $\pm$0.11	6.40 $\pm$0.03	3.1 $\pm<$ 0.01	6949 $\pm$56
QTIP (Tseng et al., 2024b)	2.036	8.38 $\pm$0.06	13.99 $\pm$0.01	14.02 $\pm$0.11	6.34 $\pm$0.02	4.0 $\pm$0.02	6400 $\pm$25
QuIP# (Tseng et al., 2024a)	2.022	8.77 $\pm$0.03	14.31 $\pm$0.03	14.25 $\pm$0.04	6.52 $\pm$0.03	1.1 $\pm$0.02	21297 $\pm$186
HIGGS (Malinovskii et al., 2025)	2.063	10.33	17.78	17.71	7.55	1.4 $\pm<$ 0.01	15223 $\pm$106
AQLM (Egiazarian et al., 2024)	2.012	7.66	13.34	12.84	5.86	75.7 $\pm$0.01	1647 $\pm<$ 1
REPS (Ours)	**2.000**	**7.20** $\pm<$ 0.01	13.33 $\pm<$ 0.01	**10.64** $\pm<$ 0.01	**5.45** $\pm<$ 0.01	**113.5** $\pm$0.33	**341** $\pm<$ 1

Quality SD summarizes three settings for KBVQ, QTIP and QuIP# (quantization randomness), and REPS (final-stage data order). Calibration data, REPS initialization and the model random seed stay fixed. Quality seed SD is not applicable to the fixed BF16 and NF4 constructions. The listed LLVQ and HIGGS endpoints also use fixed transforms. AQLM has one completed training endpoint, leaving its training variation unestimated.

Serving SD describes settings or execution orders. KBVQ, QTIP, QuIP# and REPS average load orders within each setting, then take SD across three setting means on one card per method. QuIP# uses two fixed load orders; the other three methods use four. Remaining rows report SD across execution-order means for the original payloads. Paired BF16, NF4, AQLM and REPS anchors share one A100-80GB and include all timing orders per card cohort. Appendix S2–S3 records endpoints, execution paths and cohorts.

How the representation reduces prefill cost. Replacing the LUT decoder isolates integer reconstruction's prefill cost on fixed pre-continuation Qwen3-30B-A3B weights. The two sign bytes remove codeword-table accesses from the matrix tile (Eq. 5). On one A100, TTFT falls by factors of 4.06 and 5.14 at batches one and eight. Only prefill reaches the Tensor Core path's 128-routed-pair threshold; decode is unchanged. Appendix S3.1 gives paired sequence and quality checks. Section 4.4 tests this intervention on qualified weights.

Longer generation can reverse REPS's whole-request throughput advantage (Appendix S3.2). For the original served REPS and AQLM endpoints at batch eight with 128 input tokens, output rates are 114.80/75.82 tok/s for 32 tokens and 129.05/134.35 tok/s for 512 tokens. REPS reaches the first token in about 0.34 s against 1.65 s, but AQLM has lower batch-eight inter-token latency (ITL).

Batch and prompt coverage. A 56-condition sweep of original served REPS and AQLM endpoints finds higher mean REPS output rate and lower mean TTFT throughout. Output-rate ratios range from 1.06 to 4.65 on A100 and 1.07 to 4.36 on A800; AQLM has lower mean ITL in 19 conditions. These endpoints differ in quality. Table 1 separately reports three-setting REPS means.

The sweep pairs methods within each card over four fresh-load orders per condition. Larger batches use the uncaptured path. Results stay separate by card and configuration and, with the long-output reversal, link throughput advantage to the share of request time spent in prefill.

Quality recovery in the served parent. Training recovers much of REPS and AQLM's quantization loss. Scalar post-training quantization (PTQ) loses substantial quality near two expert bpp; structured PTQ remains usable at the reported physical rates. Across 832 length-2048 windows from WikiText-2, C4, PTB and PubMed, continued REPS has lower PPL than the listed AQLM endpoint on all four corpora. It trails AQLM on the sealed 32-window development set at 10.392 against 10.289. The rows use different continuation budgets. At equal budgets, AQLM has lower PPL, so Table 1 does not establish a quantization-quality advantage for REPS. Appendix S7 gives the continuation and held-out-domain controls.

Continuation builds the served parent. REPS keeps its two-bit expert codes while updating shared non-expert tensors and the router over 8 stages and 106.8M targets. Later stages train per-matrix FP32 log-gains, folded into BF16 scales at export; the final stage includes C4 and PTB training text.

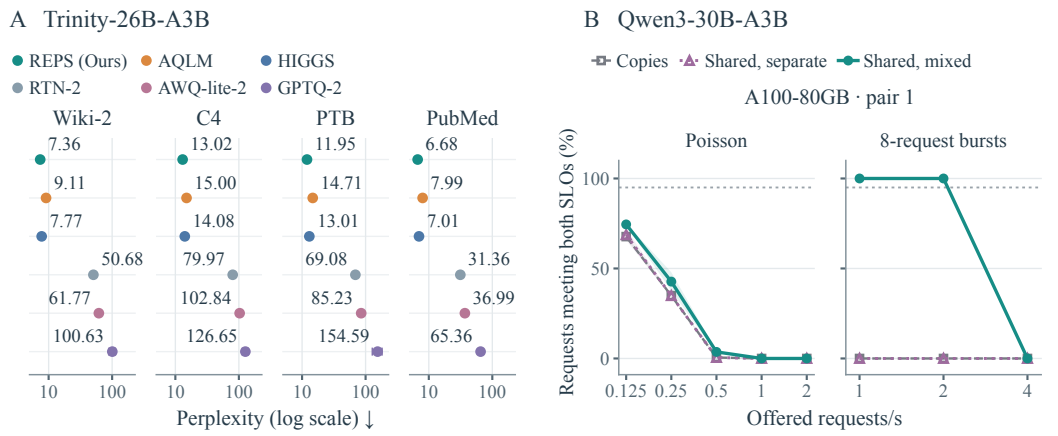


Figure 3: (A) Trinity-26B-A3B perplexity with available sample SD. (B) The first qualified pair on A100-80GB, with deployments paired on one card within each workload. Poisson curves give the mean and range of two arrival/execution orders; bursts use one. All offered requests count. Dotted lines mark 95%.

AQLM uses our QAT reproduction at step 1,500 and 49.152M targets with locally implemented code assignment.

Continuation stops at fixed budgets and uses disjoint training splits. A 13-gram blacklist filters evaluation text. Predeclared repeats retain the selected endpoint even if development loss improves.

Physical expert storage counts payloads and side information for LLVQ, KBVQ, QTIP and QuIP#. QuIP# excludes its shared runtime codebook and file container. NF4 supplies four-bit runtime context without serialized export. REPS compresses 57.98 GB of expert weights to 7.25 GB using independent projection records and a fixed decoder.

4.2 Quality on a second MoE backbone

Figure 3A compares six low-bit methods on 781 fixed windows with Trinity-26B-A3B's tokenizer. REPS and AQLM each use 500 updates under their own recipes; the other methods use PTQ. REPS has lower PPL than the other methods in panel A but trails BF16 and official four-bit NF4 Linear4bit. These PPL measurements use BF16 reconstructions.

Quality SD reflects data-order or calibration variation. REPS averages three data orders with fixed initialization, model random seed and 8,000 training windows; PPL varies by at most 0.22% across orders. GPTQ/AWQ-lite average the original first-16 calibration set and two random draws, not training seeds. Both lose substantial quality throughout. Deterministic baselines and the single AQLM endpoint have no SD.

Fixed-200 adaptation fails the joint quality requirement on Trinity. Packed and BF16 tests agree across three training seeds: REPS worsens both held-out targets, while LoRA improves them. All packed updates raise general-development perplexity. Independent cached tests find that neither method's fixed-200 updates satisfy joint target improvement and 1% retention in any of the six seed/domain cases under either policy. Exploratory development selection below retains earlier updates or parent fallbacks.

4.3 Encoding, sharing and adaptation quality

Encoding cost. Figure 4A compares encoding one selected expert per Qwen3-30B-A3B layer: 144 projections and 226.5M weights. REPS, AQLM and RTN share one A100-80GB; the other five use a second. REPS uses its no-reference encoder, AQLM one frozen-book configuration, and LLVQ, KBVQ and QTIP-HYB reference encoders. Mean encoding takes 0.314 s for REPS and 1.155 s for AQLM; RTN is faster at 0.233 s. The formats differ in quality. Timings exclude calibration, serialization, training, transfer and request serving.

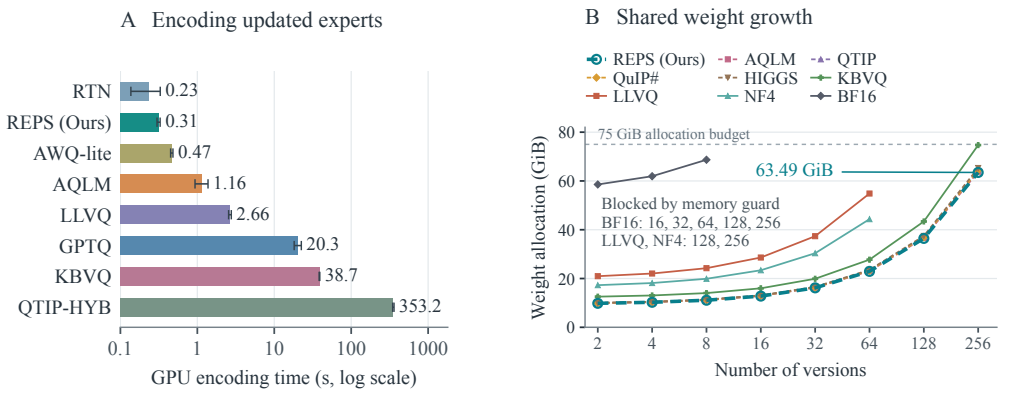


Figure 4: Qwen3-30B-A3B. (A) GPU encoding of 144 projections, mean ± SD over five runs. (B) Shared weight allocation for 2–256 structural versions, including the parent. Each added version replaces four experts per layer. Points average two same-card orders per method. The listed counts hit the 77 GiB allocator guard. Several low-bit curves nearly overlap.

Table 2: Qwen3-30B-A3B adaptation under cached mixed batching. Relative PPL change (%), mean with sample SD in percentage points over three paired data orders. Pass counts require target improvement and at most 1% degradation on each other set.

Update	Method	Target ↓	Other domain ↓	General ↓	Pass
Code	REPS (Ours)	-12.26 ±0.08	**+0.27** ±0.04	**+0.27** ±0.01	3/3
Code	LoRA (Hu et al., 2022)	**-13.83** ±0.36	+5.93 ±0.26	+5.10 ±0.05	0/3
PubMed	REPS (Ours)	**-1.00** ±0.05	**+0.27** ±0.06	**+1.64** ±0.30	0/3
PubMed	LoRA (Hu et al., 2022)	+0.14 ±0.24	+4.13 ±0.32	+8.12 ±0.42	0/3

Resident sharing. Panel B measures shared-parent allocation across nine formats from 2 to 256 structural versions. At 256 versions, REPS uses 63.49 GiB and AQLM 63.79 GiB. These measurements include decoding metadata and exclude request state, workspace and CUDA graphs. Adaptation quality and serving are evaluated separately.

Task gains and cross-domain retention. Qwen3-30B-A3B adaptation compares 200-step updates from the served parent across three paired data orders. Both methods use 4.92M targets per domain and Adafactor with constant learning rate 1.6×10^{-4}.

The two methods allocate adaptation capacity differently. REPS trains complete gate, up and down matrices in four experts per layer, selected by training-only routing counts. Scales and other parent weights stay fixed. LoRA adds rank-132 factors at each multilayer perceptron (MLP) output. REPS trains 905.97M parameters and stores 216.00 MiB of replacement tensors. LoRA uses 25.95M parameters and 49.50 MiB. Tensor sizes exclude file headers and optimizer state.

Table 2 tests target improvement and retention on 32 held-out windows per domain and 32 general-development windows, using cached mixed batching. Passing requires lower target PPL and at most 1% non-target and general PPL increases.

REPS Code combines target improvement with cross-domain retention in all three data orders. Mean non-target and general PPL each rise by 0.27%, compared with 5.93% and 5.10% for LoRA. LoRA improves target PPL more. Both methods fail the PubMed criterion, with REPS general PPL rising by 1.64%. Expert-local updates can therefore still degrade unrelated text. Pass/fail outcomes hold with partitioned batching and both full-prefill conditions. Fixed initialization makes SD a measure of data-order sensitivity. Appendix S6.1 reports all four conditions.

Earlier medical-text controls favor frozen-book AQLM and expose a REPS reload mismatch (Appendix S7). These use different endpoints.

Expert-only changes define the scope of these results (Wang et al., 2024). The released Qwen3-30B-A3B Instruct/Thinking pair also changes shared parameters, requiring a new parent. All experts

Table 3: Same-weight prefill for the separately served Qwen3-30B-A3B parent and qualified Code update. Backend time to first token (TTFT), mean ± sample SD over two execution orders on one A800-40GB. Lower is better.

Endpoint	Batch	Integer TTFT (ms) ↓	LUT TTFT (ms) ↓	LUT / integer
Parent	1	**111.02** ±0.20	388.83 ±0.40	3.50×
Parent	8	**410.59** ±0.68	1827.47 ±0.55	4.45×
Code update	1	**111.61** ±0.42	391.82 ±0.31	3.51×
Code update	8	**411.83** ±0.59	1831.33 ±0.77	4.45×

change, carrying 96.07% of difference energy (parameter-share ratio 1.01). The anonymous ledger gives per-group measurements.

4.4 Autoregressive serving of trained versions

Table 3 isolates integer versus LUT prefill on the parent and first predeclared order's fixed step-200 Code patch (Table 2). Weights, scales and decode are identical, with 128 input and 32 generated tokens. The patch improves its target within 1% retention bounds in all four scoring conditions.

Four 32-step autoregressive trajectories use independent key–value caches and each path's own argmax feedback. Full logits and tokens agree. Both endpoints pass eight-window negative log-likelihood (NLL) checks. Backend TTFT times synchronized forward and argmax, excluding queueing, input preparation and output copies.

We evaluate 105 conditions (51,120 requests) for two qualified pairs of Code and replay-trained medical patches served with their parent. All patches improve their targets within 1% retention bounds. Requests contain 128 input and 32 output tokens, with the original service-level objectives (SLOs) of 1-s TTFT and 100-ms mean inter-token interval. Figure 3B shows the first pair on A100, with same-card deployment comparisons within each workload.

In the original burst scan, mixed batches meet both SLOs for all 1,008 requests per cell at 1 and 2 requests/s in both pairs. Copies and shared, separate batches attain 0%. Mixed batches also attain 0% at 4 requests/s, with 176 and 184 unfinished requests. No Poisson cell meets all attainment, completion and backlog criteria. Appendix S6.8 separates the hardware cohorts and 27 follow-up conditions. These finite synthetic engine traces have no production frontend.

Earlier fixed-200 Qwen updates qualify only REPS Code. Its medical counterpart raises general PPL by 1.106%, and both rank-132 LoRA updates fail qualification. Across four scenarios, mixed REPS has higher mean throughput and lower mean p95 TTFT, while LoRA uses less memory. No request meets both primary SLOs in that study or its extensions. Instruction-following and free-generation quality of these earlier updates remain untested.

Trinity selection retains five patches and seven parent fallbacks across three seeds. REPS's third seed falls back in both domains. Only LoRA Code passes independent quality checks. Across 192 cells, no cell meets both p95 bounds. REPS's mean throughput is 1.08–1.72× LoRA's across the four scenarios on each platform, after averaging execution orders within each seed. Appendix S6 gives these earlier comparisons.

5 Conclusion and Limitations

REPS stores expert-local customizations as compressed replacements over one shared parent. The same blocks support independent encoding and reconstruction inside the expert operator. Sharing reduces weight allocation in structural controls. On fixed, quality-qualified weights, integer reconstruction lowers backend first-token time relative to LUT decoding.

The evidence covers one-worker language-MoE workloads. Benefits depend on adaptation quality, request length and arrival pattern. The finite burst results do not establish sustained production capacity. Shared-parameter updates require a new parent. Distributed serving, end-to-end update latency and visual or cross-modal adaptation remain untested.

AI USE STATEMENT

We used LLMs only for writing. They assisted with drafting and editing prose, with checking the manuscript for internal inconsistencies, with searching for related work, and with verifying bibliographic metadata against the original sources. The authors reviewed all AI-assisted text and take full responsibility for the content of this paper.

REFERENCES

Arcee AI. Trinity-Mini-Base model card. Hugging Face, 2025. URL https://huggingface.co/arcee-ai/Trinity-Mini-Base.

Lequn Chen, Zihao Ye, Yongji Wu, Danyang Zhuo, Luis Ceze, and Arvind Krishnamurthy. Punica: Multi-tenant LoRA serving. In *Proceedings of Machine Learning and Systems*, volume 6, pp. 1–13, 2024.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023.

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. Extreme compression of large language models via additive quantization. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Elias Frantar and Dan Alistarh. QMoE: Sub-1-bit compression of trillion parameter models. In *Proceedings of Machine Learning and Systems*, volume 6, pp. 439–451, 2024.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. OPTQ: Accurate quantization for generative pre-trained transformers. In *International Conference on Learning Representations (ICLR)*, 2023.

Hao Gu, Wei Li, Lujun Li, Qiyuan Zhu, Mark Lee, Shengjie Sun, Wei Xue, and Yike Guo. Delta decompression for MoE-based LLMs compression. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 20497–20514. PMLR, 2025.

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. In *International Conference on Learning Representations (ICLR)*, 2022.

Chi-Wei Huang and Chia-Chi Tsai. Qift: Shift-friendly no-zero W2 post-training quantization for rotated W2A4/KV4 LLM inference. *arXiv preprint arXiv:2606.02823*, 2026. URL https://arxiv.org/abs/2606.02823.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023.

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 on-device LLM compression and acceleration. In *Proceedings of Machine Learning and Systems*, volume 6, pp. 87–100, 2024.

James Liu, Guangxuan Xiao, Kai Li, Jason D. Lee, Song Han, Tri Dao, and Tianle Cai. BitDelta: Your fine-tune may only be worth one bit. In *Advances in Neural Information Processing Systems*, 2024.

Vladimir Malinovskii, Andrei Panferov, Ivan Ilin, Han Guo, Peter Richtárik, and Dan Alistarh. HIGGS: Pushing the limits of large language model quantization via the linearity theorem. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL), Volume 1: Long Papers*, 2025.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations (ICLR)*, 2017.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, Yuling Gu, Shane Arora, Akshita Bhagia, Dustin Schwenk, David Wadden, Alexander Wettig, Binyuan Hui, Tim Dettmers, Douwe Kiela, Ali Farhadi, Noah A. Smith, Pang Wei Koh, Amanpreet Singh, and Hannaneh Hajishirzi. OLMoE: Open mixture-of-experts language models. In *International Conference on Learning Representations (ICLR)*, 2025.

Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart van Baalen, and Tijmen Blankevoort. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*, 2021.

Qwen Team. Qwen3-30B-A3B-Instruct-2507 model card. Hugging Face, 2025. URL `https://huggingface.co/Qwen/Qwen3-30B-A3B-Instruct-2507`.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 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 *Proceedings of the 39th International Conference on Machine Learning (ICML)*, 2022.

Ying Sheng, Shiyi Cao, Dacheng Li, Coleman Hooper, Nicholas Lee, Shuo Yang, Christopher Chou, Banghua Zhu, Lianmin Zheng, Kurt Keutzer, Joseph E. Gonzalez, and Ion Stoica. SLoRA: Scalable serving of thousands of LoRA adapters. In *Proceedings of Machine Learning and Systems*, volume 6, pp. 296–311, 2024.

Ge Shi, Hanieh Sadri, Qian Wang, Yu Zhang, Ying Xiong, Yong Zhang, and Zhenan Fan. ExpertWeave: Efficiently serving expert-specialized fine-tuned adapters at scale. *arXiv preprint arXiv:2508.17624*, 2025.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. QuIP#: Even better LLM quantization with Hadamard incoherence and lattice codebooks. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024a.

Albert Tseng, Qingyao Sun, David Hou, and Christopher De Sa. QTIP: Quantization with trellises and incoherence processing. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024b.

Tycho F. A. van der Ouderaa, Mart van Baalen, Paul Whatmough, and Markus Nagel. Leech lattice vector quantization for efficient LLM compression. *arXiv preprint arXiv:2603.11021*, 2026. Presented at the ICML 2026 AdaptFM workshop.

Zihan Wang, Deli Chen, Damai Dai, Runxin Xu, Zhuoshu Li, and Y. Wu. Let the expert stick to his last: Expert-specialized fine-tuning for sparse architectural large language models. *arXiv preprint arXiv:2407.01906*, 2024.

Bingyang Wu, Ruidong Zhu, Zili Zhang, Peng Sun, Xuanzhe Liu, and Xin Jin. dLoRA: Dynamically orchestrating requests and adapters for LoRA LLM serving. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pp. 911–927, Santa Clara, CA, July 2024. USENIX Association.

Zukang Xu, Zhixiong Zhao, Xing Hu, Zhixuan Chen, and Dawei Yang. KBVQ-MoE: KLT-guided SVD with bias-corrected vector quantization for MoE large language models. In *International Conference on Learning Representations (ICLR)*, 2026.

Xiaozhe Yao, Qinghao Hu, and Ana Klimovic. DeltaZip: Efficient serving of multiple full-model-tuned LLMs. In *Proceedings of the Twentieth European Conference on Computer Systems*, 2025. doi: 10.1145/3689031.3717468.

Jiayu Zhao, Zihan Teng, Minhao Fan, Tianrui Ma, Wentao Ren, Song Chen, and Weichen Liu. BitsMoE: Efficient spectral energy-guided bit allocation for MoE LLM quantization. *arXiv preprint arXiv:2606.00079*, 2026.