

Lineage Is a Storage Primitive: Lossless Ancestor-Relative Storage and File-Free Serving for Fine-Tuned Model Fleets

Ibrahim Ahmed

Mohammed A. Kaleelurrahman

Abstract

A model registry’s bytes are dominated by descendants, not bases: one open 20B release has 1,235 published first-generation derivatives, 551 full-weight, and every post-training stage writes another complete checkpoint. Systems that exploit this redundancy delta against a fixed base, delta against the lineage parent lossily, or recover a base by content search, because on a public hub ancestry is rarely declared. A first-party registry knows its fine-tuning DAG exactly and for free.

We measure what that knowledge is worth over sixteen whole-model pairs and nine sampled entries from nine organisations. (i) A parent-relative delta compresses $45.03\times$ losslessly at best, against a narrow $1.43\times$ – $2.41\times$ base-relative band. (ii) In a post-training chain compressibility is a cliff at the pretraining boundary with a plateau above it, not the distance decay we pre-registered and falsified. (iii) The median changed element moves one or two ULP. (iv) Staged post-training of frontier quantised mixture-of-experts models is substantially a re-routing operation, with counterexamples.

We design ARZACH, in which the fine-tuning DAG is a registry’s physical storage layout rather than metadata, with measured content-addressed selection as the safety net for a graph that arrives dirty. Stored as a chain, three published Tulu-3-70B checkpoints occupy $2.68\times$ less than stored whole; the pairwise ratios are limits a deepening chain approaches, not its result. Such deltas serve into an unmodified inference engine — gpt-oss-20b, gpt-oss-120b and Tulu-3-8B, each a base plus one delta — reconstructed bit-exactly, hash-verified per tensor, with no file written and $25.8\times$ – $64.5\times$ fewer bytes moved.

1 Introduction

The bytes in a model registry belong overwhelmingly to derived checkpoints, not to the handful of pretrained bases they descend from. A released model is the last node of a chain rather than the output of one run: a base is supervised-fine-tuned, aligned against preferences, and often improved again

by a verifier-rewarded reinforcement-learning stage, and every one of those stages writes a complete, full-parameter checkpoint. Fleets branch as well as deepen — one open 20B release already has 1,235 published first-generation derivatives, 551 of them declaring full weights rather than an adapter or a re-quantisation — and at 70B each stage is another 141.11 GB at rest, stored whole even though it moves the median weight it touches by a single representable step.

Systems that exploit the redundancy between related checkpoints have been built for a world in which the relationship is unavailable. They delta against one fixed base, or against the lineage parent but lossily, or they recover a base by searching for a similar model — and the third choice is forced: TStore reports that of 2.75 million Hugging Face repositories only 25.8% carry family metadata at all [23]. Inverted, that statistic is this paper’s premise. A first-party registry — the storage layer of a platform that runs the training jobs whose outputs it holds — writes the lineage edge itself, when the job finishes, so the fine-tuning DAG is known exactly and for free and the question stops being *which model is this one like* and becomes *which model is this one from*.

That change of setting is worth an order of magnitude in ratio — more than any codec change we measured. Against its true lineage parent, the final RLVR stage of Tulu-3-70B compresses $45.03\times$ losslessly, whole-model, with the shipping codec (a7c-tulu70b-dpo-final.json); against a pretrained base the ratio collapses into a narrow $1.43\times$ – $2.41\times$ band across five models from three labs. Stored as a chain rather than quoted as a pair, those three published Tulu-3-70B checkpoints occupy $2.68\times$ less than stored whole — the pairwise ratio is the limit a deepening fleet approaches, never a three-model chain’s result. § 3 is the study those numbers come from, over a corpus whose lineage depth is verified rather than asserted — a column no published storage corpus carries, because it is not recoverable without the graph. Several of its results were pre-registered predictions that failed, and each is reported as a finding where it belongs.

Building on those observations we design ARZACH, in which the fine-tuning DAG is the registry’s physical storage

layout. Checkpoints are addressed by the content of their canonicalised tensor set rather than by their files, so a re-shard is not a new artefact; deltas are stored on the edges of the lineage graph and reconstructed by walking a bounded chain of them; and a serving path reconstructs a model bit-exactly from a registered base and its delta straight into an inference engine, hash-verifying every tensor after the delta has been applied, with no file written at any point. Content-addressed selection is retained as a safety net rather than a search: it repairs graphs that arrive from outside and does not out-guess a well-formed one.

Two neighbours are conceded here rather than in § 2. MGit [9] is architecturally what we build, published two years earlier — a content-addressed tensor store over a lineage graph, deltas against the parent, reconstruction by recursion up the chain — but it is lossy, and its unquantised mode expands what it stores. TStore [23] already publishes the shape of “an empirically selected base beats a fixed one”. What is ours is the combination — lossless chains in a registry that owns its lineage, whose deltas reach an engine with no file written — and a set of measurements none of these systems report.

Contributions

A measurement study of how post-training moves weights (§ 3). Four observations: parent-relative deltas beat base-relative ones by an order of magnitude; compressibility in a post-training chain is a cliff at the pretraining boundary and a plateau above it rather than a decay with distance, while distance decay is real along a pretraining trajectory; the median changed element moves one or two ULP, and that distance sets the ratio together with identity share, so neither term alone is a sufficient statistic; and staged post-training of frontier quantised mixture-of-experts models is substantially a re-routing operation, with the counterexamples shown rather than omitted. Compressibility against chain distance, MoE router-versus-expert drift and single-lineage storage all appear to be unreported; ZipNN [10] raises the base-selection question underlying the first and declines to answer it.

ARZACH, which makes lineage the physical layout of a model registry (§ 4): content addressing over canonicalised tensor sets, deltas stored on the edges of the DAG, and lossless reconstruction by walking a chain bounded at 4 hops. No prior system provides lossless multi-hop ancestor chains at a useful ratio — MGit chains, but its lossless configuration reaches $1.00\times\text{--}1.05\times$; TStore is lossless but stores every base whole; ZipLLM is lossless but flat. Its selection policy is licensed by a measurement over all 126 public full-weight derivatives of gpt-oss-20b, the entire eligible population rather than a sample.

A file-free, verified serving path (§ 5): chain $\rightarrow$ decode $\rightarrow$ per-tensor SHA-256 $\rightarrow$ inference engine, with nothing written to disk. 1,437 tensors were compared byte for

byte across two GPU classes and two weight formats with zero mismatches, and correctness is established by hashes and never by token equality.

Pre-registered negative results and published losses (§ 3, § 6.6): four design options closed off by predictions we registered against ourselves, and the regimes where the system loses — re-downloading beats us inside a datacentre, at $0.38\times$, and a warm checkpoint file loads faster than we reconstruct one.

Design principle: lineage is a storage primitive, not metadata. Prior systems record the parent in a model card and then store bytes against something else; MGit is the exception, and its lossless configuration does not pay. Here the edge the lineage names is the edge the bytes are stored on, at a ratio that does.

2 Background and related work

2.1 How weight deltas are stored today

The stages that produce a chain — supervised fine-tuning, then preference alignment by RLHF [30], DPO [33] or a verifier-rewarded method such as GRPO [35] — each write a complete, full-parameter checkpoint: unlike a LoRA adapter [12], sparse by construction, such a stage touches every tensor, and in a mixture-of-experts architecture [5] it moves routers and experts by very different amounts (§ 3). Weights ship as BF16 or quantised-integer tensors, so the natural unit of change is the ULP: the gap between adjacent representable values at a given magnitude. Systems that exploit the redundancy between such checkpoints fall into four families, separated by which base a derived checkpoint is stored against, and at what error.

(i) Against a fixed base. The dominant design deltas every derived checkpoint against one designated model, normally the pretrained base its family descends from. FM-Delta [28], the canonical lossless instance, roughly halves storage by exponent-aligned differencing; BitDelta [26] quantises the same base-relative delta to one bit per weight and DeltaZip [45] to a 2:4-sparse quantised residual, with a sizeable lossy cluster refining it [7, 16, 21, 32, 40, 44]; ZipLLM [41] applies model-aware differencing at hub scale; and single-checkpoint lossless compressors [10, 25, 39, 48] compose underneath any of them. A fixed base costs distance: a third-stage checkpoint’s delta must span every stage that produced it. ZipLLM relaxes this only when a declared base is unavailable, differencing one fine-tune against another with exact reconstruction (§4.4.4) — a nearest-*neighbour* surrogate and a one-hop repair, not a chain.

(ii) Against the lineage parent, lossily. MGit [9] is architecturally what we build: a content-addressed tensor store over an explicit lineage graph, deltas against the immediate parent, reconstruction by recursive decompression up the ancestor chain. The difference is error. MGit’s deltas are ϵ -quantised,

its own lossless configuration reaches only $1.00\times-1.05\times$ on four of its six model graphs, and its unquantised full-delta mode *expands* what it stores ($0.72\times-0.87\times$), because the difference of two full-precision tensors carries more entropy than either operand unless the codec is built for the residual. The graph-shaped store is published; a useful lossless ratio is not.

(iii) By content search, because the lineage is missing. TStore [23] and NeurStore [43] find a base by searching for a similar model, because they serve third-party hubs where ancestry is not reliably declared. Their candidate set strictly contains ours, and TStore’s result — an empirically selected base beats a fixed one, 29 % to 47 % reduction, losslessly and at tensor granularity — is the shape of our headline comparison, conceded in § 1. Inverted, though, the statistic behind their design is our premise: a first-party registry owns the graph that produced its checkpoints, so choosing a base is a walk of a known DAG rather than a nearest-neighbour query over a hub (§ 4). Work that recovers or charts lineage after the fact [11, 24, 47] treats ancestry as an object of study, not a layout.

(iv) By whole-file or chunk deduplication. Git-Theta [19], Hugging Face Xet [14], DVC [15], lakeFS [38] and Ki-tOps [17] version checkpoints as files or content-defined chunks and store what repeats verbatim. Two published facts bound that: ZipLLM measured FastCDC [42] over 43.19 TB of hub data at 14.8 % reduction and attributed even that to repeated whole tensors “rather than generic byte-level similarity”, and Git-Theta computes no delta for a dense update at all. The elements a post-training stage changes are scattered across every tensor, so essentially every 64 KiB chunk contains one and hashes differently from its parent’s, though its contents are almost entirely predictable from it.

A first-party registry removes the premise of all four: it knows the fine-tuning DAG exactly and for free, so a delta can be taken losslessly against the true parent and chained. § 3 measures what that is worth.

2.2 Serving from deltas

DeltaZip [45] is the only prior system that reconstructs a model from a delta at cold start, and it is lossy, flat and unverified, with the base already GPU-resident and the deltas read from files. TRL’s Delta Weight Sync [4] does ship *lossless* byte-level BF16 deltas into vLLM [22] — an adjacent-step hot update onto an already-running engine, deltas materialised as files, no hashing — and SkyRL’s RFC #1903 [37] adds a post-delta checksum over lossless XOR deltas for inference workers, again disk-staged and onto a warm engine. That is why we never claim to be first to verify a lossless delta for an inference engine. The no-delta streamers move whole checkpoints quickly instead: Tensorizer [3] has streamed weights without touching local disk since 2023, the Run:ai Model Streamer [29] publishes the throughput our delivery rate is

compared against, and ServerlessLLM [6] takes local materialisation as its premise [18], while adapter servers [2, 36] multiplex low-rank deltas and DeltaServe [1] co-locates inference with fine-tuning on one host. Absence of a file is not novel by itself either, and hashing during transfer is already measured and cheap [20]. Every ingredient exists separately; we are not aware of a system that combines them, which is why the claim is stated only in compound form (§ 5.2). Concurrent work on fine-grained merging for multi-model serving [8] had no preprint as this was written.

2.3 The 2026 RL-sparsity cluster

PULSE [27] derives the mechanism we measure: at post-training learning rates an Adam step is frequently smaller than the ULP at the weight’s magnitude, so roughly 99 % of per-step weight changes vanish on rounding in BF16. It, AuroraRL [34] and SparseRL-Sync [13] exploit that to compress per-step weight synchronisation, and AReaL-DTE [31] and TensorHub [46] move weights with no snapshot file at all — always within one training job, where the parent is trivially the previous step and no selection problem exists. We cite PULSE as the mechanism and claim only the extension: the movement is still this small after a whole SFT, DPO or RLVR stage, across labs that never coordinated, making it a property of *stored artefacts* rather than of a synchronisation channel.

3 Measurement study: how post-training moves weights

A registry that stores derived checkpoints as deltas is only as good as its answer to one question: *relative to what?* This section measures it on published checkpoints. Several pre-registered answers were wrong; each narrowed a claim the paper would otherwise have made, and each is reported as a finding where it belongs.

3.1 Methodology and corpus

A *diff* here is tensor-granular and lossless: checkpoints are read into a canonical form (§ 4.1), tensors matched by name, shape and dtype, and each pair sent down the cheapest codec lane (§ 4.3). Every ratio below is bytes(child)/bytes(delta), so ratios stay comparable across pairs whose parents are stored at a different width. Two instruments produce them. The *whole-model* tier runs the shipped tool over complete checkpoints on a rented pod with no sampling, and every headline number comes from it. The *sampled* tier reads fixed byte ranges of published safetensors over HTTP — 131,072 elements per tensor, class-weighted by each class’s true share of the model’s bytes — which costs nothing and makes surveys of a hundred repositories possible. It is an extrapolation and is never described otherwise; its two whole-model validation points bracket

zero and are both under 2.5 % (a7c-tulu70b-sft-dpo.json, a7c-tulu70b-dpo-final.json). Movement of individual values is reported in units in the last place (ULP): representable BF16 steps between the parent’s value and the child’s, over elements that changed at all, with cumulative mass recorded at 1 ULP, 2 ULP, 4 ULP and 8 ULP plus its median. It is not a continuous distribution and is never drawn as one. Two house rules govern the rest: no identity, sparsity or ratio figure appears without a committed raw artefact named inline, and no wall-clock figure appears without its pod class and cache state — which is why none appear here, only in § 6. Predictions were registered by commit before the runs that tested them (§ I).

Table 1 lists every stage pair with a committed raw artefact: sixteen whole-model pairs — its two Amber rows are a pretraining series, counted apart — and nine sampled entries, over twelve model families from nine publishing organisations, dense and mixture-of-experts, at five on-disk widths. No prior storage corpus reports the next two columns. *Depth* is the child’s lineage distance from its family base — verified in that every intermediate checkpoint is published and pinned by revision, so the count is a fact about artefacts rather than a claim on a model card — and *span* is the number of stages the measured pair itself crosses. Separating them is what lets the study distinguish how deep a checkpoint sits from which ancestor it was coded against.

Observation 1: Parent-relative deltas compress far better than base-relative ones.

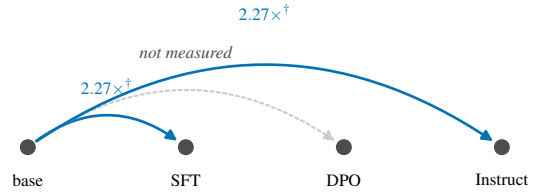
The largest instance is Tulu-3-70B’s final RLVR stage: against its DPO parent it is $45.03\times$ smaller, losslessly, whole-model (a7c-tulu70b-dpo-final.json, pre-registered at 8147380), and it is not an isolated maximum. Every whole-model pair whose parent is itself a post-trained checkpoint compresses between $3.57\times$ and $64.55\times$, with the OLMo-2-13B and OLMoE stages mid-band at $5.03\times$ – $6.16\times$; base-relative pairs occupy a narrow band around $2\times$, $1.43\times$ to $2.41\times$ over five models from three labs, dense and MoE, F16 through FP32. The separation survives organisational boundaries — Tulu descends from Meta’s Llama-3.1, Perplexity’s browsesafe from Alibaba’s Qwen3 — because what sets the ratio is whether the ancestor is a pretrained base or a post-trained checkpoint, not who published it.

A pairwise ratio is not a fleet’s storage saving, and this paper never reports one without the other. Stored as a chain the three published Tulu-3-70B checkpoints occupy 157.88 GB against 423.32 GB stored whole, $2.68\times$, because 89.4 % of what the registry holds is the single full copy at the root. What the pairwise ratios govern is the marginal cost of a child: 3.13 GB for the RLVR stage and 13.63 GB for the preference stage, against 141.11 GB for a full copy.

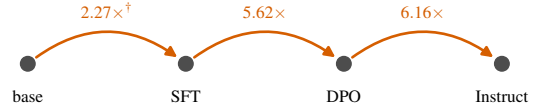
EVIDENCE: Table 1, whole-model rows; chain totals from a7c-tulu70b-sft-dpo.json and a7c-tulu70b-dpo-final.json.

Implication: The ancestor a delta is taken against is worth more than any codec change we measured, and in a first-party registry it is free to know. That is the argument for making lineage the physical

base-relative: every child against the pretrained base



stage-relative: each child against its parent (21.4 GB for three children)



† cross-dtype (FP32 base): cast-xor-zstd

Figure 1: The OLMo-2-13B chain drawn twice: against the pretrained base (left) and against each checkpoint’s immediate predecessor (right). P-G2a.1 predicted monotone decay and was falsified. The base-relative edges are flat in distance, $2.273\times$ at $d=1$ and $2.272\times$ at $d=3$. They are cross-dtype (FP32 base, BF16 derivatives), but the cast lane is not what makes them small: re-diffed same-dtype the ratio is unchanged to six figures. base→DPO was never measured and is drawn dotted, not interpolated. g2a-figdata.json, g2b-olmorne-olmosft.json.

layout rather than a metadata field (§ 4.2), and for reporting fleet savings as marginal cost per child (§ 6.3).

Observation 2: Compressibility is a cliff at the pretraining boundary, not a decay with distance.

We pre-registered the opposite (P-G2a.1, ccccd306) — that the ratio would fall monotonically with the number of stages between parent and child, with the falsifier stated in advance — and it failed. On the OLMo-2-13B chain the $d=1$ pair base→SFT compresses $2.273\times$, below the $d=2$ pair SFT→Instruct at $5.041\times$. Distance is not the variable.

What the five whole-model rungs support instead is a two-regime law (Fig. 1). Against the pretrained base the ratio collapses to $2.27\times$ and is flat in distance: base→SFT is $2.273\times$ and base→Instruct, two further stages later, is $2.272\times$ — itself 3.3 % above the $1.5\times$ – $2.2\times$ band P-G2a.3 registered for it, which we score a second miss — and mean relative drift agrees from the other side, 1.833×10^{-2} against 1.836×10^{-2} . Against a post-training ancestor the ratio is $5\times$ – $6\times$ and does decay gently with span — $6.163\times$ and $5.622\times$ at span 1 against $5.041\times$ at span 2, the skip-stage pair sitting below both steps it skips. Choosing the nearest post-training ancestor over the base is worth $2.2\times$ – $2.7\times$; choosing it over a more distant one, only $1.1\times$ – $1.2\times$.

The cliff is clean drift and not an artefact of the cast lane those two base-relative pairs run — A12 ships this base at FP32 and every derivative at BF16. Re-diffed against a BF16 rendition of the same base on the plain xor-zstd lane, base→SFT returns $2.273005\times$ against the cast lane’s $2.273005\times$, a difference of 2,102 bytes in 12.07 GB (g2b-olmorne-olmosft.json). It is measured on one family, and base→DPO was not measured; the base-relative rows of Table 1 for four other models land in the same $1.43\times$ – $2.41\times$

Table 1: The measurement corpus and the compression result in one table. D and S are the depth and span columns of § 3.1; *child* is the size of the checkpoint being stored, which is the numerator of the ratio beside it. Every ARZACH row is a whole checkpoint, lossless, no sampling. The INTDIFF column re-encodes the same delta with the candidate integer-difference lane, shown as the resulting ratio (the underlying measured gains are *byte* reductions of 5.59 %–11.52 %); it is a measured candidate, not the shipped codec. Baseline rows are our runs of those baselines’ own code on the same bytes, stated as bytes needed for the child given its parent is already stored. Artefact names omit .json. For the gpt-oss families the pretrained base is unpublished, so depth is counted from the released checkpoint, itself post-trained. The MoE families are OLMoE, gpt-oss, browsesafe/Qwen3, JetMoE and DeepSeek-V3; the rest are dense.

Family (publisher)	Stage pair	D	S	dtype	Child (GB)	Ratio	INTDIFF	Artefact
<i>ARZACH — whole-model tier, lossless, our runs</i>								
gpt-oss-120b (OpenAI)	release→safeguard	1	1	MXFP4	65.25	64.55×	—	a10-gptoss120-safeguard
Tulu-3-70B (AI2)	DPO→final (RLVR)	3	1	BF16	141.11	45.03×	47.70×	a7c-tulu70b-dpo-final
gpt-oss-20b (OpenAI)	release→safeguard	1	1	MXFP4	13.76	25.79×	28.52×	a10-gptoss20-safeguard
Tulu-3-70B (AI2)	SFT→DPO	2	1	BF16	141.11	10.35×	11.34×	a7c-tulu70b-sft-dpo
Tulu-3-8B (AI2)	DPO→final	3	1	BF16	16.06	7.23×	—	a7c-tulu8b-dpo-final
OLMo-2-13B (AI2)	DPO→Instruct	3	1	BF16	27.43	6.16×	—	g2a-olmo13b-dpo-instruct
OLMo-2-13B (AI2)	SFT→DPO	2	1	BF16	27.43	5.62×	—	g2a-olmo13b-sft-dpo
OLMo-2-13B (AI2)	SFT→Instruct	3	2	BF16	27.43	5.04×	—	g2a-olmo13b-sft-instruct
OLMoE-0924 (AI2)	SFT→Instruct	2	1	BF16	13.84	5.03×	5.69×	a9-olmoe0924-sft-inst
browsesafe (Pplx.)	Instruct→browsesafe	2	1	BF16	61.06	3.57×	—	a10-browsesafe
Amber (IFM)*	adjacent pretrain ckpts	—	1	BF16	13.48	2.79×	—	g2b-a200-a201
Qwen2.5-72B (Qwen)	base→Instruct	1	1	BF16	145.41	2.41×	—	f1b-72b
OLMo-2-13B (AI2)	base→SFT†	1	1	FP32→BF16	27.43	2.27×	—	g2a-olmo13b-base-sft
OLMo-2-13B (AI2)	base→Instruct†	3	3	FP32→BF16	27.43	2.27×	—	g2a-olmo13b-base-instruct
OLMoE-0924 (AI2)	base→Instruct	2	2	BF16	13.84	1.85×	—	f1a-olmoe-drift
Qwen2.5-0.5B (Qwen)	base→Instruct	1	1	BF16	0.99	1.81×	—	f1a-qwen05b
Vicuna-7B (LMSYS)‡	base→Vicuna	1	1	F16	13.48	1.43×	—	a4-deltazip-vicuna7b
Amber (IFM)*	$d=200$ pretrain ckpts	—	—	BF16	13.48	1.39×	—	g2b-a000-a200
<i>Lossless baselines on the same bytes — our runs, Tulu-3-70B DPO→final^{fl}</i>								
ZipNN [10] single-model				BF16	141.11	1.51×	—	g1-baselines
zstd -3, raw bytes [§]				BF16	141.11	1.29×	—	g1-baselines
FastCDC [42] dedup				BF16	141.11	1.0007×	—	g1-baselines
ZipNN root + our delta				BF16	141.11	2.92×	—	g1-baselines
<i>Lossy systems in their own regime — published, not our runs</i>								
DeltaZip [45] 4-bit, 2:4	Llama-2-7B→Vicuna	1	1	F16	13.48	4×–11×	—	published
BitDelta [26], 1-bit					—	not run	—	—

Sampled tier — nine further rows over eight families (Tulu-3/3.1 counted as one), Table 3

*A pretraining series, not a fine-tuning stage, so depth from a family base is undefined; both rows are BF16 on both sides, and Amber’s FP32 checkpoints are excluded (§ H). S counts checkpoint intervals for these two rows: one for the adjacent pair, 200 for the other, which is a step count rather than a stage count and is not comparable with the fine-tuning rows, so it is left undefined. $d=200$ is the largest separation the release allows — 133 % mean relative drift, mean cosine 0.47 — so 1.39× is what the instrument returns when no lineage signal is left, and a ratio there is not evidence of ancestry. †Cast lane (FP32 parent, BF16 child); re-diffed same-dtype the ratio is unchanged to six figures, so it carries no dtype caveat (§ 3). ‡DeltaZip’s own pair, where we lose: full-model tune at F16, which carries three more mantissa bits than BF16 into the planes the codec compresses, and lossless cannot beat 4-bit with 50 % structured sparsity (§ 6.6). §At -19 zstd reaches 1.32×; the band registered at ccccd306 was 1.1×–1.3×, and the default level is what “plain zstd” means in practice. Published figures for the same methods, for contrast: ZipNN 1.49×, Xet/CDC dedup 1.17×, MGit’s lossless mode 1.00×–1.05×, its unquantised full-delta mode 0.72×–0.87× (it expands). ¶Lossless delta systems not run here, and why: FM-Delta [28] has no BF16 path and this corpus is almost entirely BF16; MGit [9] quantises its deltas by default, so only its lossless configuration is comparable and that is the published figure quoted above; ZipLLM [41] is flat, and the chunk-dedup regime it measures is the FastCDC row we did run on these bytes; TStore [23] recovers a base by content search, which is the question § 4.4 measures directly on this corpus. All four are conceded by name in § 2.

band.

Distance decay is not dead everywhere — it is a pretraining law. On IFM/Amber, a 1.26T-token pretraining series of 360 checkpoints evenly spaced at about 3.5B tokens, the same prediction’s shape clause holds: whole-model and BF16 on both sides throughout, the ratio falls monotonically from 2.794× at $d=1$ to 1.389× at $d=200$ with no inversion (§ H). Both registered *levels* missed (P-G2b.1, ccccd306) — adjacent checkpoints compress 2.79× against a predicted 1.5×–2.0×, and $d=128$ gives 1.49× against a predicted below 1.2×, a bound never reached even at $d=200$ — so the prediction missed on levels and held on shape. What the ratio tracks is accumulated drift rather than checkpoint count: sorted by drift the same pairs stay monotone with forward and backward partners interleaved, while at matched separation the two directions

disagree sharply — 2.451× at 59 forward against 1.676× at 64 backward — because training slows down. One pretraining step moves the weights about 35× further than a whole DPO stage does (2.9×10^{-2} against 8.3×10^{-4} mean relative drift, across two model families, so an order-of-magnitude statement rather than a controlled comparison). And 1.389× is the floor — what the instrument returns when no lineage signal is left (133 % drift, mean cosine 0.47) — so a ratio there is not evidence of ancestry.

EVIDENCE: The five g2a-olmo13b-*.json rungs of Table 1, g2a-figdata.json, g2b-olmorne-olmosft.json and the eleven Amber pairs of g2b-figdata.json (§ H); predictions at ccccd306.

Implication: Selection should reach for the nearest *post-training* ancestor and is close to indifferent among the rest: most of the available win is captured by not falling back to the base. An $O(\text{depth})$

walk of a known lineage executes that policy exactly (§ 4.4).

Observation 3: The changed elements move one or two ULP, and that distance sets the ratio together with identity share.

PULSE derives this mechanism for a single optimiser step [27]; what we measure is its *per-stage* residue, across labs and architectures. Over the eleven sampled pairs that carry a ULP distribution — six of Table 3’s rows plus five Tulu-3-70B and Tulu-3-8B pairs the sampled table does not list separately — the median changed element moves one or two ULP, and in nine of the eleven it is exactly 1 ULP in every class sampled. The mass is concentrated as well as centred: 63.9 %–66.2 % of changed elements sit within 1 ULP on the Tulu-3-70B pairs and 93.5 %–94.0 % within 8 ULP.

The two exceptions matter more than the nine agreements: zephyr-7b-beta medians 2 ULP in all three sampled classes with only 40.8 % of its changed elements within 1 ULP, and OLMoE-1B-7B-0125 medians 2 ULP in its embedding class. Finer slices are rougher still — a layer-5 attention slice of the OLMo-2-13B DPO→Instruct pair medians 3 ULP against 1 ULP for the same class at layer 20 — so the claim is the narrow one, not the uniform version. The signature also belongs to post-training stages rather than to fine-tuning in general: on base-relative pairs the same instrument reports medians of 14 ULP and 19 ULP in the byte-dominant expert class (a5-codec-bakeoff.json), the pairs that compress 1.81× and 1.72× on that instrument. Stage matters within post-training too: on Tulu-3-70B, DPO moves the weights about 20× further than the subsequent RLVR stage (0.0278 % against 0.0014 % mean relative drift).

The ratio is identity share *plus* that ULP distance, not identity alone. If unchanged elements were free and changed ones cost their full width, a pair whose elements are a fraction p identical would compress by exactly $1/(1-p)$. That bound is wrong in both directions. It over-predicts the most identical pair measured — Tulu-3-70B DPO→final is 98.2 % element-identical in its mlp class, bounding the ratio at 55.6× against a measured 44.37×, because at that sparsity the code spends its bytes saying *where* the changes are rather than what they are — and every less identical pair beats it: the measured ratio is 1.19 times the bound at 88.7 % identity and 2.32 times it at 24.1 %, because elements that moved a single step cost far less than their width.

That last pair is zephyr, and it is what makes the two-term reading necessary. zephyr retains only 24.1 % element identity and still reaches 3.06×, while OLMoE base→Instruct retains 2.5 % in its experts and reaches 1.81× on the same instrument (a5-codec-bakeoff.json): a tenfold difference in identity buys under 1.7× in ratio. Yet zephyr is also the lowest-ratio post-training pair we measured, and the pair whose changed mass sits furthest out — the outlier under both terms, which is what a two-term law predicts and a one-term law cannot express. Whole-*tensor* identity substitutes for neither: the OLMo-2-13B post-training pairs have one bit-identical tensor in 443 and still compress 5×–6×.

EVIDENCE: a7-dense-chains-sampled.json,
a8-crosslab-sampled.json, a9-moe-sampled.json,
a5-codec-bakeoff.json, g2a-olmo13b-sft-dpo.json
and the two Tulu-3-70B artefacts.

Implication: An element that moved one representable step XORs

to a residual with a single low bit set, which is why a plane-split XOR followed by a general entropy coder is the right shape for the shipped lane (§ 4.3) and an explicit element-level bitmap is not; an integer-difference variant buys a further 6 %–12 % of delta bytes (§ 6.3). And a selection policy must score candidates on both terms: identity fraction is not a sufficient statistic, and neither is a similarity metric that reduces a candidate to one distance — which is what a store recovering a base by nearest-neighbour search is obliged to use (§ 4.4).

Observation 4: Staged post-training of frontier quantised MoEs is substantially a re-routing operation.

On both gpt-oss pairs the router tensors move an order of magnitude further than the expert tensors they gate. The metric is mean relative drift, not the fraction of weights that changed, which is $\approx 100\%$ in both classes of every pair: 0.68 % against 0.046 % on the 20b pair and 0.74 % against 0.052 % on the 120b, ratios of 15.0× and 14.3× (a10-gptoss20-safeguard.json, a10-gptoss120-safeguard.json). Both artefacts come from one lab and both are MXFP4, so the claim is scoped to staged post-training of frontier quantised MoEs and no wider; the counterexamples belong in the same paragraph. On OLMoE-1B-7B-0924 SFT→Instruct routers and experts move indistinguishably, 0.169 % against 0.170 % (a9-olmoe0924-sft-inst.json); on the same family’s base→Instruct pair the ordering reverses, 5.78 % against 7.35 % (fla-olmoe-drift.json), as it does on browsesafe, 0.53 % against 0.62 % (a10-browsesafe.json). OLMoE’s load-balancing loss gives every expert gradient [5], which would explain the parity; we did not measure the routing regime, so this is a conjecture. DeepSeek-V3-Base→R1-Zero points the same way as the gpt-oss pairs but under a different instrument — router 5.75 against expert 1.61 compressed bits per element, on an FP8 pair whose endpoints were quantised independently — and is never averaged into the drift figures above. The storage consequence runs the other way: routers are under 0.05 % of an MoE checkpoint’s bytes and experts 78 %–93 % (a9-moe-sampled.json), so the artefact’s byte cost is set almost entirely by how far the experts moved.

EVIDENCE: All five artefacts named inline; DeepSeek from a5-codec-bakeoff.json.

Implication: Lane choice must be made per tensor and not per model: in the gpt-oss pairs the expert tensors carry the bytes and barely move while the routers move and cost nothing, and a policy averaged over the model would mis-serve both.

Negative result: Four pre-registered results that closed off a design option.

- **The element-level bitmap codec is not worth building.** We registered, against ourselves, that an explicit bitmap-plus-values encoding would *not* beat an LZMA-coded XOR by more than 20 % (8b70b32). It held — the gain is 6 %–12 % of *ratio*, on the sampled tier — and it killed the largest piece of engineering we had proposed (a5-codec-bakeoff.json).
- **τ -reversion is bounded-error, not lossless.** We predicted that reverting sub-threshold elements to the parent’s value would be exact, reasoning that those elements had not moved (8b70b32). They had — by one representable step, which is Observation 3, and at FP8’s three mantissa bits 1 ULP is

$\approx 12.5\%$ relative error. The τ -tier ratios are never quoted as lossless and are used nowhere in this paper.

- **Row granularity is the wrong grain.** We registered that at least 25 % of vocabulary rows would be bit-identical across the RLVR stage (3c7e816). On `embed_tokens` at 70B it held (43.9 % of a sampled region); on `lm_head` it failed outright, 0.0 %–3.1 % of rows against 98.2 %–98.8 % of *elements* (a7-dense-chains-sampled.json). The softmax gives every output row gradient, so a row codec collects nothing the element lane does not.
- **Positioned parallel writes are slower.** Registered at $1.6\times$ or better, measured at $0.87\times$ in a same-pod interleaved A/B (d32051b): the reconstruction path is bound by total disk traffic, not by the writer.

Seven further predictions registered at ccccd306 did not survive their runs and are reported where they belong: monotone distance decay, the base→Instruct band and both level clauses of the pre-training ladder (Observation 2); declared-parent sufficiency and the bound on what selection leaves behind (§ 4.4); the integer-difference lane’s lower bound, which held on one pair and missed low on another; and the verification-overhead bound, which missed on its registered pod (§ 6.5).

Each observation’s limits are stated where it is made; § 7 collects the ones that bound the paper’s claims rather than a single result.

4 Design: the registry

In ARZACH the fine-tuning DAG is the registry’s physical storage layout. Each observation licenses one mechanism: storing every artefact as a delta against its lineage parent (§ 4.2), a residual-plus-entropy-coder codec rather than a sparsity structure (§ 4.3), and a selection policy that prefers the nearest declared ancestor and then verifies rather than trusts it (§ 4.4). Underneath all three sits one correctness obligation — a delta means nothing except relative to bytes that can be reproduced exactly — so the registry addresses tensor sets by content and canonicalises their serialisation before anything is diffed (§ 4.1). Figure 2, in the appendix, draws the whole of it.

4.1 Content addressing and canonical form

What is addressed is the tensor set, not the file. A checkpoint’s identity is a SHA-256 over its per-tensor hashes taken in name order, where each per-tensor hash covers the tensor’s name, dtype, shape and raw little-endian bytes; a pairwise Merkle reduction over the same sorted leaves yields a root that later supports partial verification. Two properties of that definition point in opposite directions. *Layout is invisible*: shard count, boundaries and tensor order are not in the hash, so a three-shard checkpoint and its single-file save are one identity and a delta computed against one applies unchanged to the other.

Precision is not invisible, on purpose: dtype is fed to the hasher, so a BF16 and an F16 save of the same weights are two bases. Dropping it would be the wrong fix, because casting is directional and the rule that made re-saves collapse would also declare an F32 master and its degraded BF16 release one model. What identity gives up here the codec recovers — an exact re-save into another float format is a different identity and a zero-byte delta (§ 4.3).

Canonicalisation is therefore a correctness obligation rather than an optimisation. Ingest streams a checkpoint, however sharded, into one canonical single-file layout — name-sorted, deterministic header — so the same tensor set always produces identical bytes; if that held only approximately, a delta would be applied to bytes other than those it was computed against, which the per-tensor post-hash would turn into a refusal rather than a wrong model, leaving the chain unusable. We measure the property rather than assume it (§ 6.2). Blobs live in a content-addressed object store keyed by SHA-256 and written under a temporary name, so an interrupted push never leaves a partial object under its final name; deduplication is a consequence of the addressing, not a feature.

4.2 Delta chains as physical layout

A manifest names its reference by content hash, and nothing else. Resolving that hash yields either a registered base — memory-mapped in place — or another artefact, rebuilt from *its* reference; the two meet in one recursive resolution step and the delta arithmetic is identical either way — the whole of “storage follows the DAG” on the read side. When the reference is another artefact the manifest also records the storage edge explicitly, with the hop count back to a registered base; depth 0 is a keyframe, reconstructible without walking anything.

The cost is linear in depth, and we bound it at write time rather than hoping. Chains are limited to 4 hops: a push whose parent already sits at the limit is encoded against the registered base instead, and the API response says so, so a caller who asked for a parent and got a keyframe knows which they have. Resolution refuses a manifest claiming a deeper chain rather than following it. The limit is small on purpose: the ratios that motivate chains come from *one* well-chosen parent, and every additional hop is another full reconstruction of a multi-gigabyte tensor set — at least the 37.3 s–42.9 s an in-memory reconstruction of a 65.2 GB artefact takes on an H100 PCIe pod with the pack page-cache warm (g4-stream-gptoss120.json), and more where the intermediate hop is materialised rather than streamed.

This is the paper’s design principle in one sentence: **lineage is a storage primitive, not metadata**. Prior systems record the parent in a card and then delta against a fixed base [26, 28, 41] or search for a similar one [23, 43]; here the edge the lineage names *is* the edge the bytes are stored on.

4.3 Codec

The shipped codec is the XORZSTD family, applied per tensor and chosen per tensor by measurement, in order of preference. `identity` costs zero bytes; `scale-id` costs one f64 and is kept for a tensor whose parent differs from it by a single uniform factor — a case we designed for and never once observed: the detector fires in no pair we measured, and a scaled residual never beat plain XOR (`fla-olmoe-drift.json`); `cast-id` likewise costs nothing when the custom tensor is exactly the base cast into another float format. The residual lanes below them — `scale-xor-zstd`, `cast-xor-zstd` and `xor-zstd` — XOR against a prediction, split the residual into byte planes and compress with zstd, and tensors absent from the reference fall back to a whole-tensor `raw-zstd` blob. The manifest is simultaneously the payload index and the sparsity map, since an `identity` entry carries no blob at all. Every tensor carries a class inferred from its path in the module tree: nothing here is compressed lossily, but the classification would keep a future lossy lane away from routers, norms, embeddings and the LM head.

A measured candidate. XOR is the wrong distance function for a checkpoint delta: a weight that moved one representable step — the modal post-training update — has an XOR residual whose magnitude depends on where the binary carry landed, so the entropy coder ends up modelling carries rather than training. INTDIFF maps each float onto an order-preserving integer key and subtracts, through four stages that are each a bijection on the raw bit pattern, before handing the result to the same plane-split-and-zstd pipeline (§ D). It gains 6 %–12 % in delta bytes whole-model (§ 6.3) and is nonetheless *opt-in and not the shipped default*: every published ratio in this paper was produced by the xor lane, so flipping the default would make those figures unattributable. A decoder that does not know the identifier rejects it outright, which is what the pure-Python loader would do today.

4.4 Ancestor selection

The policy is: take the lineage-declared parent, then verify it. In a first-party registry that edge is written by the registry at push time rather than by a model card, so the candidate set is a path and the walk is $O(\text{depth})$ — at most the 4 hops of § 4.2.

Whether the declared graph is enough is an empirical question, and we pre-registered an answer that was wrong: that the declared parent would be the best available delta base for at least 70 % of derivatives (P-G5.1, ccccd306). Measured over all 126 public full-weight derivatives of `gpt-oss-20b` — the entire eligible population, so there is no sampling frame to defend, though every ratio behind it is a sampled-tier extrapolation — it is best for 39.7 %, and always coding against it leaves 19.1 % of delta bytes on the table, an extrapolated

197.5 GB of 1,034.2 GB. That second figure falsifies a registration of its own: P-G5.3 put the aggregate at $\leq 10\%$. Both figures assume the reading most favourable to lineage, in which the declared parent may be converted into the child’s tensor space for free; under the strict reading they are 7.1 % and 62.3 %.

The cause is not mistaken ancestry but format hygiene: 112 of the 126 derivatives are published in a tensor space their declared parent is not in, because the base ships MXFP4 and every fine-tuning stack writes BF16 back out. Where parent and child are byte-comparable at all the declared parent wins outright for every subject that is not an exact re-upload (6 of 6), the median margin where content selection wins is 8.2 % — P-G5.2 registered $\leq 25\%$, which holds charitably and fails at 73.6 % strictly — and the declared parent is within 1 % of the best candidate for 52.4 % of subjects (`g5-selection-survey.json`). Two of the survey’s three registered predictions fail outright. Three failure modes of the *public* graph account for the gap, and the design answers each.

Format hygiene is answered by canonicalising on ingest and carrying the cast lanes in the codec, so a dtype re-save is a zero-byte delta rather than an incomparable artefact. *Duplication* — 73.2 GB of the 197.5 GB that selection saves — is collected by a content-addressed store under either policy, so counting it as selection’s win would flatter the mechanism. *Hidden ancestors* are the case only content search can make, and the survey contains one (§ E).

The resulting claim is deliberately narrow. First-party registries have trustworthy declared graphs, because the registry writes them; there lineage is both the correct answer and the cheap one, and content-addressed selection is the safety net for what arrives from outside. It repairs a broken declared graph; it does not out-guess a well-formed one, and the survey’s own limits (§ 7) make the benefit it measures a lower bound.

Why a path rather than a search. A content search over a large enough index can only match or beat a path on ratio, so the argument for the $O(\text{depth})$ walk is cost and determinism rather than reachable ratios. It is *index-free* — at most four manifest reads, with no vector index to build, tune or defend a recall parameter for; *deterministic* — the same push produces the same edge, so an artefact’s bytes are reproducible from its manifest alone, which an index cannot promise across re-indexing; and *available before the child exists*, since the parent is known at push time and the delta is computed once, on the write path. The honest limit is that public data barely exercises it: of the 1,235 first-generation derivatives of `gpt-oss-20b`, roughly ten second-generation artefacts are measurable at all (`g5-selection-survey.json`). The deep chains this argument is about are a property of first-party fleets, the setting this paper is written for.

5 Serving from the chain

A registry that stores chains is only useful if what it stores can be served. ARZACH plugs into vLLM as a load format: the engine is pointed at a directory holding only a model’s configuration and tokeniser, and the weights arrive from the registry as a registered base plus the delta encoding the child against it, reconstructed and verified in host memory on their way in. Deeper ancestors are resolved by the daemon first (§ 4.2); every serving measurement here is a base and one delta. Two decoder implementations share one manifest contract, pure-Python and Rust; every measurement below uses the Rust decoder, which decodes in parallel.

5.1 The stream path

Follow one tensor. The loader reads the pack’s manifest and, from it, the content hash of the reference the pack was encoded against, which resolves to a registered base on the local filesystem, memory-mapped in place. A pack whose reference is itself an artefact needs that ancestor resolved by the daemon first (§ 4.2), which materialises it, so the file-free path described here is the last edge of a chain rather than the whole of one. Off the registry host the pure-Python decoder instead fetches the base blob once per node into a local cache and gates it on its own SHA-256, which costs nothing to check because the blob’s identifier *is* its hash. Nothing is unpacked to disk: the loader’s download hook is deliberately a no-op.

Reconstruction then runs as a pool of decode workers over the manifest, which push completed tensors to the calling Python thread *in whatever order they finish*: vLLM’s weight loading accepts any order, and dropping the constraint is what lets decode saturate the cores while Python copies an earlier tensor to the device. The decoded buffer reaches Python through the CPython buffer protocol over the worker’s own allocation, so the tensor the engine sees is a view of the bytes the decoder produced rather than a copy, and vLLM performs the host-to-device copy itself (§ C).

The path is defined as much by what it never creates. No model file, no delta file, no checkpoint directory and no temporary of any kind is written on this path: the bytes exist as a mapping of the base, a mapping or buffer of the delta blob, and one reconstructed tensor per in-flight worker.

Peak host memory is bounded by construction: a worker reserves a tensor’s estimated cost against a global budget *before* decoding it, so peak is that budget plus a few finished tensors rather than a function of model size (§ C).

5.2 Verification and corruption refusal

Every tensor carries a SHA-256 of its *reconstructed* bytes in the manifest, and the loader recomputes it over the bytes it just produced — after the delta has been applied, not over what came off the wire. That is the difference from the published

precedent: the OCI cold-start study [20] hashes container bytes in transit and reports that doing so costs under 0.1 %, but hashing in transit says the compressed delta arrived intact, which is weaker than saying the weights that came out of the decoder are the weights the manifest promised. The hash is fused into the decode pass rather than taken as a second sweep — each completed 2 MB block is fed to the hasher while it is still in cache, and the block callback must cover the output exactly once, in order — because a separate pass over a reconstructed 60 GB tensor set would be a second cold traversal of everything.

Refusal semantics are absolute and local: a tensor whose recomputed hash does not match its manifest entry is never handed to the sink. The decode returns an error naming the tensor and its codec, the first error stops every worker, and the failure surfaces as an exception inside the engine’s weight-loading call, so the engine aborts a load rather than completing one with a wrong tensor in it. An unrecognised codec identifier is likewise a hard error — an old decoder meeting a newer artefact refuses it.

This is the paper’s serving claim, stated once and in full: a full-precision model is reconstructed bit-exactly from a lossless delta against a content-addressed base and streamed straight into an inference engine, hash-verified per tensor in flight, with no model, delta or checkpoint file materialised at any point. Each conjunct alone has prior art (§ 2.2); the conjunction appears to be new.

5.3 The correctness instrument

Correctness is established by bytes, never by tokens: every streamed tensor’s SHA-256 is compared against the same tensor’s bytes in a checkpoint materialised from the same pack, and whole-file hashes are compared across pods and builds. We do not compare generated tokens, for an empirical reason — both serving campaigns observed vLLM’s greedy decoding flip a near-tie at the same token position between two runs of the *same* loader, so token comparison has a noise floor of its own and proves strictly less than byte equality, which has none (a14-vllm-stream-gptoss20.json, a15-stream-opt-gptoss20.json).

5.4 Breadth, and what the path costs

The path is not a single-artefact result. Four runs have exercised it on three artefacts spanning 13.8 GB to 65.2 GB, two GPU classes and two weight formats — MXFP4 mixture-of-experts and dense BF16 — and the three runs that record per-tensor byte equality compared 1,437 tensors with zero mismatches (§ 6.2). The BF16 case matters most for scope: it was the first non-MXFP4 model put through the loader, served on the first attempt with no code change, and matched the file path byte for byte, so the claim covers dense models as well as quantised MoEs.

The quantity we lead with is bytes moved, because it is a property of the method rather than of the box: serving from the registry moves the pack, not the checkpoint — 0.534 GB instead of 13.76 GB on the 20B artefact and 1.011 GB instead of 65.25 GB on the 120B, a reduction of $25.8\times$ and $64.5\times$ (a15-stream-opt-gptoss20.json, g4-stream-gptoss120.json). Wall-clock supports that and does not lead it: streaming beats materialise-then-load by $1.72\times$ – $2.84\times$ across the four interleaved A/Bs of Table 2. Two caveats travel with those numbers wherever they appear. First, the margin is a *disk-crossing* argument: the $2.84\times$ is against a materialise running at 0.39 GB s^{-1} , and against the fastest materialise we have ever recorded for the same pair (1.59 GB s^{-1} on an L4 pod, a13-speedrace-gptoss120.json) the file route composes to roughly 106s and the stream path wins only $1.30\times$. Second, an earlier 22.37s measurement of the 20B stream path did not reproduce on its own pod class: with engine arguments pinned, the same pack and harness measured 40.61s–44.17s there (g4-verify-ab-gptoss20.json). The discrepancy is not in the loader — across the two pod classes weight loading is 4s–7s of a 30s–44s construct — but in engine initialisation, which the earlier run did not record. We report both and average neither.

Verification is real work whose cost depends entirely on what it overlaps: unoverlapped it is large, 31.7% on the engine’s own weight-load stage, while overlapped — the serving case — it sits at or below each pod’s noise floor. What makes per-tensor verification nearly free at serving time is overlap, not cheapness (§ 6.5).

6 Evaluation

6.1 Setup

Every number here is whole-model — complete checkpoints diffed with the shipped tool and codec — unless it says otherwise; all 126 sampled-tier numbers were later re-measured from committed raw JSON, 122 agreeing within 1%. Runs used rented single-node pods (H100 PCIe, A100 PCIe, L4, RTX 4090, RTX 4000 Ada), configured as in § B. Arms alternate within one pod session so anything making the box slower moves both; the first round is discarded, because the first vLLM start on a fresh pod pays JIT and compile costs no later start pays; the statistic is min-of-3, every round in the committed JSON. Every wall-clock number below carries its pod class and cache state, and packs stay page-cache warm after the push that created them — the condition a serving host runs in. Where a number did not reproduce we say so at the point of use rather than averaging it away.

6.2 Correctness

The instrument is § 5.3’s: SHA-256 over each *reconstructed* tensor’s bytes against the same tensor in an independently materialised checkpoint, plus whole-file hashes across pods and builds. Byte equality has no noise floor, which is why we report it and not token agreement. The 1,437 comparisons decompose as 687 on gpt-oss-120b (H100 PCIe, g4-stream-gptoss120.json), 459 on gpt-oss-20b (A100 PCIe, g4-secondpod-gptoss20.json) and 291 on Tulu-3-8B (A100 PCIe, g4-bf16-tulu8b.json) — zero mismatched and zero missing in each of those three. A fourth run drove the same 459 gpt-oss-20b tensors through both arms of the verification A/B (g4-verify-ab-gptoss20.json). Reconstruction is also deterministic across machines and binaries: 12 of 12 materialise runs produced byte-identical whole-file SHA-256 across two pod classes and three separately compiled binaries (a13-speedrace-gptoss120.json, a13-speedrace2-gptoss20.json). A codec whose output depends on the host is not a storage primitive; this is how we know ours does not.

6.3 Compression versus baselines

Table 1 is the whole-model picture, corpus and baselines in one table. The headline $45.03\times$ was pre-registered before the run and the $10.35\times$ one stage earlier was registered at ccd306 to land in $9\times$ – $13\times$; both are pairwise, and the chain they sit on reduces by $2.68\times$ (Observation 1). The weakest same-dtype parent-relative ratio on a *post-training* pair is $3.57\times$, on a third-party derivative of a Qwen3 MoE that moved nearly every expert weight; the two Amber rows are Observation 2’s pretraining regime.

The candidate codec. Re-diffing all four whole-model pairs through the integer-difference lane (g7-rediff-report.md) improves the delta by 5.59%–11.52% in *bytes*, which is 5.9%–13.0% in *ratio* units; we quote bytes throughout and say so, because on these four pairs the two differ by 0.3–1.5 points, and by about 1.7 on the sampled slice. The gain is inversely related to identity share — DPO→final, the most identical pair, gains least; OLMoE-0924, the least identical, gains most — because INTDIFF improves the coding of *changed* elements. It is therefore **6%–12%, inversely related to identity share** and never flat, which is why P-G7.1 (8%–30% on two named pairs) is scored a **partial miss**: OLMoE held, Tulu DPO→final missed low. P-G7.2 (0%–15% on gpt-oss) held at 9.56%. The lane is a measured candidate; the shipped codec is XORZSTD.

Baselines are our runs of their code, not their published numbers. Every lossless baseline in Table 1 was run on the same volume as our own ratios, so nothing is quoted second-hand (g1-baselines.json), and all three pre-registered pre-

Table 2: Top: the interleaved A/B, min-of-3, with the stream arm’s range over the three scored rounds bracketed (the file arm is far steadier — every round within 9 % of its minimum). *File* is materialise plus the default loader’s construct, both fresh in the same round; all arms page-cache warm, warm-up discarded; gpt-oss rows are MXFP4 MoE, Tulu-3-8B dense BF16. The fifth row is not a separate run but the same 120b stream re-composed against the fastest materialise on record (1.59 GB s^{-1} , L4), the bound we treat as binding. Bottom: verification cost on the same instrument, where one environment variable disables the post-hash and nothing else. Per-round detail is in § B (g4-stream-gptoss120.json, g4-secondpod-gptoss20.json, g4-bf16-tulu8b.json, g4-verify-ab-gptoss20.json, a15-stream-opt-gptoss20.json).

Artefact (GB)	Pod	Stream (s)	File (s)	Speed-up
gpt-oss-20b (13.76) [†]	H100	22.37 [22–23]	55.31	2.47×
gpt-oss-120b (65.25)	H100	82.15 [82–139]	233.08	2.84×
gpt-oss-20b (13.76)	A100	29.97 [30–49]	51.62	1.72×
Tulu-3-8B (16.06)	A100	20.81 [21–38]	46.33	2.23×
gpt-oss-120b (65.25)	mixed	82.15	~106	1.30×
<i>Verification overhead, same-pod A/B, min / median</i>				
vLLM construct, 20b*	H100		3.18 % / 6.40 %	
vLLM construct, 20b	A100		0.24 % / 0.30 %	
weight-load stage, 20b	H100		31.7 % / 31.7 %	
standalone drain, 20b	H100		15.6 % / 29.3 %	
standalone drain, 120b	H100		64.3 % / 67.0 %	

*The row the registered prediction applies to. [†]From an earlier campaign (a15-stream-opt-gptoss20.json); it did not reproduce on its own pod class, which measured 40.61 s–44.17 s with engine arguments pinned (§ 5.4).

dictions held: ZipNN reaches $1.51\times$ (P-G1.1, registered $1.3\times$ – $1.7\times$), plain zstd $1.29\times$ at its default level (P-G1.2, registered $1.1\times$ – $1.3\times$) — we disclose that at -19 it reaches $1.32\times$, just outside the band, rather than rounding the ceiling into it — and FastCDC-style chunking recovers 95 MB across both checkpoints (282 GB), a ratio of $1.00034\times$ (P-G1.3, registered $<1.2\times$, and not narrowly); Table 1’s $1.0007\times$ is the same run against the child alone. A generic binary differ finds much of the same redundancy — xdelta3 -9 reaches $14.53\times$ against our lane’s $40.49\times$ on one matched shard (§ F) — so what the tensor-aware step buys is the rest of the ratio, cheaply. ZipNN is complementary rather than a rival: it compresses the one root copy and we remove the children, which is why a ZipNN-compressed root plus our delta is the best row in the table at $2.92\times$. The lossy GPU tier is quoted as published, not re-run.

6.4 Time to serving

Table 2 is the A/B in full; § 5.4 states what the margins mean and the caveats that travel with them, and § B has the per-round evidence and the three serving pre-registration verdicts. What the stream path removes is one stage: the file write is 167.6 s of the 120b file route’s 233.1 s, while the stream reconstructs the same 687 tensors, hash-verified, in 42.5 s.

Delivery rate. Decode, verify and deliver runs at 2.1 GB s^{-1} on the gpt-oss-20b pack (H100 PCIe, page-cache warm, a15-stream-opt-gptoss20.json), which ties a tuned streamer’s published local tier (2) and *loses* to its tuned S3 tier (3.1), both as that vendor reports them [29]. Parity on rate while moving $25.8\times$ – $64.5\times$ fewer bytes is the claim; a rate win is not.

Reconstruction cost is per-byte, not per-model. End-to-end reconstruct holds at approximately 1.6 GB s^{-1} from 13.76 GB to 65.25 GB — 1.67 GB s^{-1} and 1.592 GB s^{-1} , a -4.7% change across $4.74\times$ in size — with 12/12 runs byte-identical across pods and builds (a13-speedrace2-gptoss20.json, a13-speedrace-gptoss120.json). Those two points come from two different pod classes, so what is demonstrated is invariance across size *and* hardware, which is weaker than a controlled sweep. The dense BF16 xor lane is in fact the cheapest to reconstruct in the campaign, which suggests 1.6 GB s^{-1} is a floor set by the MXFP4 block layout rather than by the design.

6.5 Verification overhead

The lower block of Table 2 is the measurement § 5.2 promises. Pre-registered prediction P-G8.1 was $\leq 3\%$ of stream wall-clock (cccd306); at 3.18 % on the minimum and 6.40 % on the median it **missed on the H100 pod** and held on the second; the registration fixed the interleaving and the run count but named no pod class, so we report it per pod rather than pooling the two into a pass. On that pod the engine’s run-to-run spread was 8.5 s against a gap of 1.25 s, so three runs per arm establish no more than “a few percent, inside engine noise” (§ B). The last three rows are the same cost with the overlap removed: verification costs about 1.6 s on 13.76 GB, which is 32 % of a weight-load stage that is itself 5 s–7 s of a 40 s–48 s construct on that pod. The honest figure once the overlap is gone is 32 %, not 3 %.

6.6 Where we lose

Inside a datacentre, re-downloading is faster. On a pod with a 625 MB s^{-1} path to the model hub, fetching the whole 13.76 GB checkpoint took 22.01 s while the delta path — fetch pack, decode, verify, write — took 57.24 s. That is $0.38\times$: we lose, and by more than the pre-registered losing range allowed (a11-coldstart-gptoss20.json). From the same numbers the crossover is at about 235 MB s^{-1} , below which the delta path wins ($8.28\times$ at 20 MB s^{-1}). What does not depend on link bandwidth is the $25.8\times$ reduction in bytes moved, which is the egress, storage and replication bill rather than the stopwatch.

A warm file loads faster than we reconstruct. On the same pod, vLLM’s default loader reads an already-

materialised checkpoint in 3.6s–4.0s against our 7.38s–7.55s, because in that window we decode, hash-verify and deliver rather than mmap bytes that are already correct on disk (a15-stream-opt-gptoss20.json). The loss is real. The reply is that the file has to be written before it can be read: 34.7s of materialise from the same 0.534 GB pack our 7.38s starts from, against the 13.76 GB file the default loader needs on disk first.

Lossy systems win on ratio in their own regime.

On DeltaZip’s own pair — Llama-2-7B to Vicuna-7B, a full-model tune — our lossless ratio is $1.43\times$, our lowest on record, against their published $4\times\text{--}11\times$ [45] (a4-deltazip-vicuna7b.json). The gap is not closable by engineering: lossless cannot beat 4-bit weights with 50% structured sparsity. Two things sharpen rather than excuse it. First, both checkpoints are F16, which carries three more mantissa bits than BF16 into exactly the byte planes the codec compresses: a doubly unfavourable regime, in which our registered $1.8\times\text{--}2.5\times$ was falsified low and our prediction that some tensors would be identical was falsified outright — none was. Second, the comparison is not like-for-like: their number is a lossy delta against a fixed base, ours an exact reconstruction.

7 Discussion and limitations

Public chains proxy enterprise fleets. Every number here comes from published checkpoints; no customer artefact was measured. Chain structure and weight statistics should carry over, because the SFT→preference→RL sequence a lab publishes is the same operation a platform’s tenants run; access patterns, retention policy and fleet size do not, and the deep chains the $O(\text{depth})$ argument is about barely exist in public (§ 4.4), so that part of the design rests on mechanism and cost rather than on a public measurement. Two instrument limits bound the study the same way: Observation 2’s cliff is measured on one model family and its pretraining contrast on one other, and the ULP distribution is recorded at four cumulative points rather than element-wise, so Observation 3 constrains the median and the mass out to 8 ULP and nothing past it.

Two things the storage cost depends on that are not training. The first was a confound we registered against ourselves and then closed: at pre-registration the two base-relative rungs of Observation 2 were also the only cross-dtype pairs, so $2.27\times$ was disclosed as a lower bound, and the same-dtype re-diff has since shown the cast lane contributed nothing to it. The same run established the second, which is a claim rather than a caveat — *dtype is a first-class term in what a lineage costs*: at fixed, measured training distance one Amber checkpoint step costs 0.72 delta bytes per parameter in BF16 and 2.91 in FP32, $4.05\times$ for the same weight movement

(§ H.1), so a registry’s bill is set as much by how its inputs are written as by how much they were trained. The publisher can also get the export wrong outright — Amber’s first 260 checkpoints ship every RMSNorm weight at the initialisation value rather than the trained one (§ H.2), which is why this paper quotes no whole-tensor identity rate for that family. That is § 4.4’s dirty declared graph again in another form: a failure of *publication*, and an argument for a registry that records artefacts as they are produced.

Two confounds we could not remove. Quantisation confounds the MoE result in the direction that supports it: both gpt-oss pairs are MXFP4 and from one lab, the DeepSeek pair is FP8 with independently quantised endpoints, and neither instrument separates the quantisation from the drift — which is why the claim is scoped to staged post-training of frontier quantised MoEs and the counterexamples at parity and reversed are reported rather than dropped. And wall-clock is a property of the pod as much as of the loader: one stream measurement (22.37s on the 20B artefact) did not reproduce on its own pod class, where the same pack and harness with engine arguments pinned gave 40.61s–44.17s. We report both and average neither, treat only ratios measured inside one interleaved session as sound, and lead with bytes moved.

Where the approach loses, and why there is no lossy tier.

The two serving losses are measured in § 6.6; on storage, the regimes to expect little from are the ones the study names — any delta that crosses the pretraining boundary, and F16 checkpoints, which carry three more mantissa bits than BF16 into exactly the byte planes the codec compresses. A lossy tier would buy ratio, and plainly: on DeltaZip’s own pair we reach $1.43\times$ against their published $4\times\text{--}11\times$. We refused one for a reason we measured rather than assumed: our own pre-registered τ -reversion proposal was falsified by Observation 3, because an element that moved a single representable step has moved, and at FP8’s three mantissa bits that step is $\approx 12.5\%$ relative error. A registry is also the system of record, and an artefact that cannot be reproduced bit-for-bit is not the artefact that was trained. The selection survey of § 4.4 is narrow for its own reasons: sampled tier, one base model whose MXFP4→BF16 break dominates the result, and “nearest” meaning nearest within a 127-model index rather than a ranked retrieval over a hub — so the benefit it measures is a lower bound rather than an asymptote.

Future work. The pretraining ladder of Observation 2 is one release whose forward distances are capped at 59 by its own format switch; a second series, ideally one spaced by weight movement rather than by tokens, would separate the decay’s rate from the schedule that produced it. Separately, storing checkpoints on their lineage edges makes checkpoint repair tractable: substituting or interpolating tensors between

two checkpoints of one lineage becomes a registry operation once the edges are where the bytes live.

8 Conclusion

Lineage is a storage primitive, not metadata. A registry that knows which training job produced which checkpoint — and a first-party registry always does — can store a derived checkpoint’s bytes on the edge the lineage names, rather than recording that edge in a card and coding the bytes elsewhere. Stored as a chain, the three published Tulu-3-70B post-training checkpoints occupy $2.68\times$ less than stored whole, a reduction that climbs with fleet depth towards the asymptotes its parent-relative deltas set. Chains of that kind serve into an unmodified inference engine — gpt-oss-20b, gpt-oss-120b and Tulu-3-8B, each a registered base plus one delta — reconstructed bit-exactly and hash-verified tensor by tensor, with no file written anywhere and $25.8\times\text{--}64.5\times$ fewer bytes moved. Reaching those numbers cost us several pre-registered predictions, each reported where it belongs and each released with its registration SHA (<https://example.org/artifact>).

References

- [1] Jiaxuan Chen, Jianshu She, Ye Yuan, Rajat Ghosh, Karan Gupta, Qirong Ho, Xue Liu, and Oana Balmau. DeltaServe: Host-Agnostic Co-Serving of inference and Fine-Tuning for LLMs. *arXiv preprint arXiv:2607.28848*, 2026.
- [2] Lequn Chen, Zihao Ye, Yongji Wu, Danyang Zhuo, Luis Ceze, and Arvind Krishnamurthy. Punica: Multi-Tenant LoRA serving. *arXiv preprint arXiv:2310.18547*, 2023.
- [3] CoreWeave. Tensorizer: Fast model serialization and streaming. <https://github.com/coreweave/tensorizer>, 2023. Accessed 2026-08-27.
- [4] Amine Dirhoussi, Quentin Gallouédec, Kashif Rasul, Lewis Tunstall, Edward Beeching, and Albert Villanova del Moral. Shipping a trillion parameters with a hub bucket: Delta weight sync in TRL. Hugging Face blog, <https://huggingface.co/blog/delta-weight-sync>, May 2026. Published 27 May 2026. Accessed 2026-08-27.
- [5] 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.
- [6] Yao Fu, Leyang Xue, Yeqi Huang, Andrei-Octavian Brabete, Dmitrii Ustiugov, Yuvraj Patel, and Luo Mai. ServerlessLLM: Low-Latency serverless inference for large language models. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2024.
- [7] Hao Gu, Wei Li, Lujun Li, Qiyuan Zhu, Mark Lee, Shengjie Sun, Wei Xue, and Yike Guo. Delta decompression for MoE-based LLMs compression. *arXiv preprint arXiv:2502.17298*, 2025.
- [8] Vima Gupta, Oytun Kудay Duran, Nandan Suresh Meda, Ikhyun An, Ganesh Ananthanarayanan, and Anand Iyer. SANDHI: Fine-grained merging for memory efficient Multi-Model serving. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, 2026. To appear; no preprint available as of 2026-08-27.
- [9] Wei Hao, Daniel Mendoza, Rafael da Silva, Deepak Narayanan, and Amar Phanishaye. MGit: A model versioning and management system. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.
- [10] Moshik Hershcovitch, Andrew Wood, Leshem Choshen, Guy Girmonsky, Roy Leibovitz, Ilias Ennmouri, Michal Malka, Peter Chin, Swaminathan Sundararaman, and Danny Harnik. ZipNN: Lossless compression for AI models. *arXiv preprint arXiv:2411.05239*, 2024.
- [11] Eliahu Horwitz, Nitzan Kurer, Jonathan Kahana, Liel Amar, and Yedid Hoshen. We should chart an atlas of all the world’s models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.
- [12] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank adaptation of large language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.
- [13] Lucas Hu, Ranchi Zhao, Isaac Zhu, Zach Zhang, Hscos Zhang, Hugh Yin, and Jason Zhao. SparseRL-Sync: Lossless weight synchronization with $\sim 100\times$ less communication. *arXiv preprint arXiv:2605.07330*, 2026.
- [14] Hugging Face. Xet: Chunk-level deduplication for the Hub. <https://huggingface.co/docs/hub/en/storage-backends>, 2025. Accessed 2026-08-27.
- [15] Iterative, Inc. DVC: Data version control. <https://dvc.org>, 2026. Accessed 2026-08-27.
- [16] Yanfeng Jiang, Zelan Yang, Bohua Chen, Shen Li, Yong Li, and Tao Li. DeltaDQ: Ultra-high delta compression for fine-tuned LLMs via group-wise dropout and separate quantization. *arXiv preprint arXiv:2410.08666*, 2024.
- [17] Jozu. KitOps: Packaging and versioning AI/ML projects as OCI artifacts. <https://kitops.org>, 2026. Accessed 2026-08-27.
- [18] Huzaifa Shaaban Kabakibo, Animesh Trivedi, and Lin Wang. Breaking the ice: Analyzing cold start latency in vLLM. *arXiv preprint arXiv:2606.07362*, 2026.
- [19] Nikhil Kandpal, Brian Lester, Mohammed Muqeeth, Anisha Mascarenhas, Monty Evans, Vishal Baskaran, Tenghao Huang, Haokun Liu, and Colin Raffel. Git-Theta: A git extension for collaborative development of machine learning models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023.
- [20] Georgii Kliukovkin. Cold-Start model delivery in Kubernetes inference serving: An empirical study of OCI-Based distribution and its integrity. *arXiv preprint arXiv:2607.16596*, 2026.
- [21] Stefan Kuyumdzhev and Radostin Cholakov. Per-Axis weight deltas for frequent model updates. *arXiv preprint arXiv:2512.19720*, 2025.
- [22] 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 ACM Symposium on Operating Systems Principles (SOSP)*, 2023.
- [23] Tingfeng Lan, Zirui Wang, Yunjia Zheng, Zhaoyuan Su, Juncheng Yang, and Yue Cheng. TStore: Rethinking AI model hub with Tensor-Centric compression. *arXiv preprint arXiv:2604.17104*, 2026.
- [24] Benjamin Laufer, Hamidah Oderinwale, and Jon Kleinberg. Anatomy of a machine learning ecosystem: 2 million models on Hugging Face. *arXiv preprint arXiv:2508.06811*, 2025.
- [25] Wenshuo Li, Xinghao Chen, Han Shu, Yehui Tang, and Yunhe Wang. ExCP: Extreme LLM checkpoint compression via Weight-Momentum joint shrinking. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.
- [26] 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 (NeurIPS)*, 2024.
- [27] Erfan Miahhi and Eugene Belilovsky. Understanding and exploiting weight update sparsity for Communication-Efficient distributed RL. *arXiv preprint arXiv:2602.03839*, 2026.

- [28] Wanyi Ning, Jingyu Wang, Qi Qi, Mengde Zhu, Haifeng Sun, Daixuan Cheng, Jianxin Liao, and Ce Zhang. FM-Delta: Lossless compression for storing massive fine-tuned foundation models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pages 66796–66825. Curran Associates, Inc., 2024.
- [29] NVIDIA Run:ai. Run:ai model streamer: Performance benchmarks. <https://github.com/run-ai/runai-model-streamer>, 2025. Accessed 2026-08-27.
- [30] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- [31] Yingqi Peng, Jiawei Zhang, Wenhao Zhou, Ruida Xu, Ran Yan, Wei Dong, Yi Gao, Zhiqiang Ding, Tongkai Yang, and Binhang Yuan. AReaL-DTE: Sparse Policy-Weight transfer for online agentic reinforcement learning. arXiv preprint arXiv:2608.00455, 2026.
- [32] Bowen Ping, Shuo Wang, Hanqing Wang, Xu Han, Yuzhuang Xu, Yukun Yan, Yun Chen, Baobao Chang, Zhiyuan Liu, and Maosong Sun. Delta-CoMe: Training-Free Delta-Compression with Mixed-Precision for large language models. arXiv preprint arXiv:2406.08903, 2024.
- [33] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.
- [34] Chaoyi Ruan, Geng Luo, Xinyi Wan, Long Zhao, Qinghe Wang, Jiaan Zhu, Duling Xu, Guanbin Xu, Dehui Wei, Xiang Liu, Cheng Li, Haifeng Sun, Liang Luo, Congcong Miao, and Jialin Li. AuroraRL: Fast, Fault-Tolerant, and Cost-Efficient reinforcement learning over decentralized network. arXiv preprint arXiv:2602.11456, 2026.
- [35] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, HaoWei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. arXiv preprint arXiv:2402.03300, 2024.
- [36] 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. S-LoRA: Serving thousands of concurrent LoRA adapters. arXiv preprint arXiv:2311.03285, 2023.
- [37] SkyRL Contributors. [RFC] delta weight sync in SkyRL. NovaSky-AI/SkyRL issue #1903, <https://github.com/NovaSky-AI/SkyRL/issues/1903>, July 2026. Accessed 2026-08-27.
- [38] Treeverse. lakeFS: Data version control for object storage. <https://lakefs.io>, 2026. Accessed 2026-08-27.
- [39] Daniel Waddington and Cornel Constantinescu. Lossless compression for LLM tensor incremental snapshots. arXiv preprint arXiv:2505.09810, 2025.
- [40] Xiaohui Wang, Peng Ye, Chenyu Huang, Shenghe Zheng, Bo Zhang, Lei Bai, Wanli Ouyang, and Tao Chen. Breaking the compression ceiling: Data-Free pipeline for Ultra-Efficient delta compression. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. The method is named UltraDelta.
- [41] Zirui Wang, Tingfeng Lan, Zhaoyuan Su, Juncheng Yang, and Yue Cheng. ZipLLM: Efficient LLM storage via Model-Aware synergistic data deduplication and compression. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2026.
- [42] Wen Xia, Yukun Zhou, Hong Jiang, Dan Feng, Yu Hua, Yuchong Hu, Qing Liu, and Yucheng Zhang. FastCDC: A fast and efficient content-defined chunking approach for data deduplication. In *Proceedings of the USENIX Annual Technical Conference (ATC)*, pages 101–114, 2016.
- [43] Siqi Xiang, Sheng Wang, Xiaokui Xiao, Cong Yue, Zhanhao Zhao, and Beng Chin Ooi. NeurStore: Efficient in-database deep learning model management system. arXiv preprint arXiv:2509.03228, 2025.
- [44] Yan Yang, Yixia Li, Hongru Wang, Xuetao Wei, Jianqiao Yu, Yun Chen, and Guanhua Chen. ImPart: Importance-Aware Delta-Sparsification for improved model compression and merging in LLMs. arXiv preprint arXiv:2504.13237, 2025.
- [45] Xiaozhe Yao, Qinghao Hu, and Ana Klimovic. DeltaZip: Efficient serving of multiple Full-Model-Tuned LLMs. In *Proceedings of the European Conference on Computer Systems (EuroSys)*, 2025.
- [46] Chenhao Ye, Huaizheng Zhang, Mingcong Han, Baoquan Zhong, Xiang Li, Qixiang Chen, Xinyi Zhang, Weidong Zhang, Kaihua Jiang, Wang Zhang, He Sun, Wencong Xiao, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. TensorHub: Scalable and elastic weight transfer for LLM RL training. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, 2026.
- [47] Runpeng Yu and Xinchao Wang. Neural lineage. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4797–4807, 2024.
- [48] Tianyi Zhang, Mohsen Hariri, Shaochen Zhong, Vipin Chaudhary, Yang Sui, Xia Hu, and Anshumali Shrivastava. 70% size, 100% accuracy: Lossless LLM compression for efficient GPU inference via Dynamic-Length float (DFloat11). arXiv preprint arXiv:2504.11651, 2025.

A System diagram

Fig. 2 draws the three mechanisms of § 4 in one picture: canonicalising ingest, deltas held on the edges of the lineage DAG, and the serving path that walks a chain into an engine without writing a file.

B Per-round serving measurements

Engine configuration for every serving measurement: vLLM 0.11.0, transformers 4.57.6, max_model_len 4,096, gpu_memory_utilization 0.90 (0.97 at 120b, whose weights alone are 76 % of the device), on H100 PCIe 80 GB (128 vCPU) and A100 PCIe 80 GB (252 vCPU) pods. Table 2 reports min-of-3 with the stream arm’s range. The rounds behind it, and the two facts that license reading the difference as the loader rather than the box, are here.

The spread is engine init, not the loader. Round 1 of the 120b stream is 138.58 s, which reads as decoder warm-up and is not: its *engine-init* stage — profile run, KV allocation, CUDA-graph capture, none of it the loader’s work — took 80.11 s, against 30 s–34 s in rounds 2–3 and 27.19 s–28.54 s in every file round. Once that one-time cost is paid the two arms’ engine init agree within a few seconds, and that agreement is what licenses reading the rest of the difference as the loader. It also explains the cross-pod gap: the A100 beat the H100 PCIe pod by 26 % on the same pack in the same week, with a modestly faster loader stage (4.26 s–5.04 s against 5.22 s–6.97 s) but a much faster engine init.

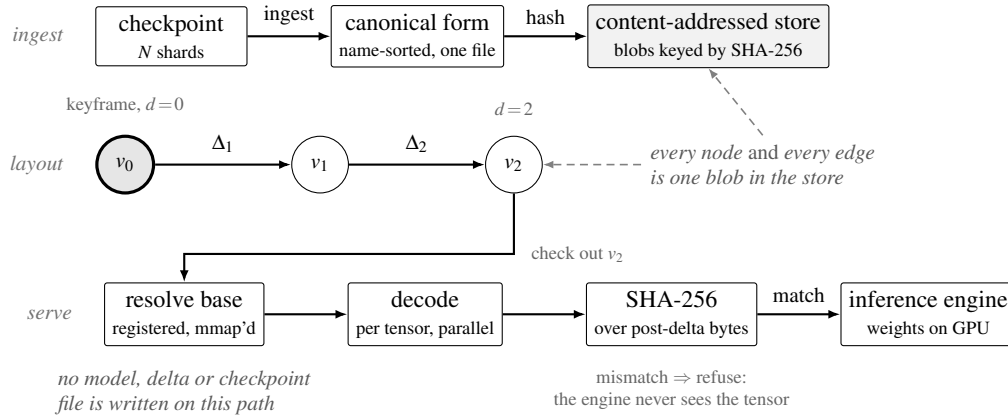


Figure 2: ARZACH. Ingest canonicalises a checkpoint into a name-sorted tensor set and addresses it by the hash of that set, so shard layout is invisible to identity. Deltas are stored *on the edges* of the lineage DAG: a manifest names its reference by content hash, and the daemon rebuilds an artefact whose reference is itself an artefact by recursion (at most 4 hops). The serving lane shown here is the last edge of such a chain: it resolves a registered base, decodes each tensor, verifies its SHA-256 against the manifest and hands the buffer to the engine, writing no file of any kind.

The file arm is writer-bound, not device-bound. Three measurements say so, and they are why the margin is a disk-crossing argument rather than a decoder result. `dd` with `O_DIRECT` does 1.7 GB s^{-1} each way on the 120b pod’s container disk; a single-file materialise took 165.97 s, indistinguishable from the sharded runs, so layout is not the cost; and a standalone drain — identical decode, identical verification, no file written — takes 37.25 s–42.86 s. The roughly 125 s between drain and materialise is the write path moving 65 GB at about 0.5 GB s^{-1} against a device that does 1.7 GB s^{-1} .

Serving pre-registration verdicts. Three predictions were registered at `cccd306` before any pod existed. P-G4.1 (120b stream in 45 s–90 s and $\geq 1.5\times$ versus file, same pod) **held on both halves**. P-G4.2 (BF16 dense lane bit-exact, <35 s) **held**, and its falsifier — failure would have narrowed the serving claim to quantised MoE artefacts — did not fire. P-G4.3 asked whether the 20b path would repeat on a second pod class within a registered tolerance of an earlier measurement, and **held**; its confound matters more than its verdict. That earlier measurement did not reproduce on its own pod class (§ 5.4), so the tolerance was scored against a number we could not re-measure. The prediction is recorded as held and the framing is retired: no claim in this paper rests on it.

Reading the verification miss with its spread. Verify-OFF constructs on the registered pod ranged 39.36 s–47.88 s — 8.5 s of engine variance — while the ON-versus-OFF gap at the minimum is 1.25 s. Three runs per arm cannot establish more than “a few percent, inside engine noise”; tightening that needs more runs, not a better argument. The unoverlapped rows of Table 2 reconcile it: verification costs about 1.6 s on 13.76 GB, which is 32 % on a weight-load stage that is itself

5 s–7 s of a 40 s–48 s construct on this pod. (The wider 4 s–7 s of 30 s–44 s quoted in § 5.4 pools both pod classes.)

C Buffer lifetime on the serving path

Workers claim contiguous runs of tensors sized by bytes rather than by count (256 MB per run), so a run of many small tensors and a run holding one large tensor cost a worker about the same, and completed tensors travel to the caller through a bounded channel. The ordered pipeline still exists and remains the default for writing a file, where ordering makes the writes sequential.

Before a worker decodes a tensor it reserves that tensor’s estimated cost — output bytes plus the decompressed residual — against a global budget, blocking until the budget allows. The budget defaults to 8 GiB, is overridable, and is never permitted to fall below twice the largest single-tensor estimate, so a tensor larger than the budget can always proceed alone and the cap cannot deadlock. Reservations are released as soon as a tensor is handed on, so with the bounded channel of § 5.1 added, peak host memory is the budget plus a few finished tensors. Reserving *before* decoding rather than counting buffers afterwards is what makes that true: on the 120B artefact the transient to control was dozens of workers each holding a multi-gigabyte expert tensor mid-decode, which no ready-buffer accounting can see. A reconstructed buffer’s lifetime ends when the engine drops its reference; nothing model-sized is retained across tensors.

D The integer-difference candidate lane

INTDIFF maps each float onto an order-preserving integer key, subtracts with wraparound, zigzags, and hands the result to

the same plane-split-and-zstd pipeline the xor lanes use. Each of the four stages is a bijection on the raw bit pattern and no stage inspects a value, so losslessness does not depend on the interpretation being right. Even when the lane is enabled it stays a candidate: it displaces the xor blob only when strictly smaller and the zero-byte tiers still outrank both, so enabling it cannot enlarge any delta. It is not the shipped default because every published ratio in this paper was produced by the xor lane and because the inference-engine loader accepts only the three xor lanes, rejecting an unknown codec identifier outright, so the default could not move before the loader’s parity work lands.

E Two details from the selection survey

Where a quantised format has to be undone before a declared parent can be used, the conversion is deterministic and anchored: our dequantisation of the MXFP4 `gpt-oss-20b` root reproduces an independently published BF16 conversion of it byte for byte over a 16.6 MB probe, zero differing bytes (`g5-selection-survey.json`). And the survey’s cleanest single case for content selection is a hidden ancestor: four F16 repositories from different uploaders have expert, router and embedding tensors byte-identical to each other and differ only in attention, so they descend from an intermediate checkpoint that was never published and appears in no declared graph. Content search finds them ($27\times$ – $77\times$ against each other); the declared graph cannot contain them.

F Isolating the tensor-aware step

The lossless baselines of [Table 1](#) answer “what does another system store?”. A second comparison answers “how much of our ratio is the delta and how much is knowing the tensor layout?”. On one matched 4.66 GB shard of the Tulu-3-70B DPO→final pair, same machine and same run: `zstd --patch-from` gives $1.29\times$, `xdelta3 -9` gives $14.53\times$, and the xor-plus-zstd core of our lane gives $40.49\times$ at $1,989\text{ MB s}^{-1}$ (`g1-baselines.json`). A generic binary differ finds most of the redundancy; knowing where the tensors begin is what makes the rest of it cheap. Storing both Tulu-3-70B checkpoints costs 219.52 GB under zstd and 187.02 GB under ZipNN, against 144.24 GB for parent-whole plus our delta and 96.64 GB for a ZipNN-compressed root plus our delta.

G Sampled-tier corpus rows

[Table 1](#) carries the whole-model tier, from which every headline number comes. The sampled tier — ranged reads, 131,072 elements per tensor, class-weighted, an extrapolation rather than a measurement of the whole checkpoint — adds the following, and the ranges are over the layers sampled.

All rows are BF16 except DeepSeek-R1-Zero (FP8) and the selection survey (mixed; its subjects span MXFP4, BF16 and F16). The publishers here are AI2, HuggingFace H4, MyShell, DeepSeek, Alibaba and OpenAI; the three that appear nowhere in [Table 1](#) are what take the corpus from six publishing organisations to nine. The survey is the instrument behind § 4.4: a fixed slice per model, weighted by exact per-class byte shares, over a 127-model index, which is why only its per-subject comparisons are treated as robust.

H The Amber pretraining ladder

Observation 2’s pretraining contrast is [Table 4](#): an anchor at `ckpt_200` of IFM/Amber diffed against ten other checkpoints of the same run, BF16 on both sides throughout so `full_bytes` is 13.48 GB in every row and the ratios are directly comparable. Distance is signed: positive is later in training.

The ordering is monotone with no inversion, and monotone within each direction separately. It is also monotone in mean relative drift with forward and backward partners interleaved, which is the reading that transfers: checkpoint distance predicts the ratio only through drift, and it is a poor proxy across training phases because training slows down. The forward half of the ladder is capped at $d = 59$ by Amber’s own format switch, so no single decay constant should be quoted from this table.

H.1 Controls at fixed distance

Amber publishes `ckpt_000`–`ckpt_259` in BF16 (13.48 GB) and `ckpt_260`–`ckpt_358` in FP32 (26.95 GB). A distance study that did not check would have published a spurious cliff at $d = 64$ that is a file format. [Table 5](#) holds distance fixed and varies format instead.

Read the drift column first: at $d = 1$ all three pairs moved the weights by essentially the same amount, so the training effect is indistinguishable. The ratio is $2.794\times$ in BF16 and $1.38\times$ in FP32. Three things follow. It is not the boundary crossing but the width: $259\rightarrow 260$ (crosses formats) and $260\rightarrow 261$ (entirely inside FP32) differ by 0.1 %. Distance still works inside the FP32 stratum, the whole curve merely shifted down. And the same training step costs $4.05\times$ more delta bytes per parameter when the checkpoint is published at FP32 — the measurement behind § 7’s claim that dtype is a first-class term in what a lineage costs.

H.2 The untrained norms

In every BF16 pair above, including `ckpt_000` against `ckpt_200` 700B tokens apart, all 65 RMSNorm weight tensors are bit-identical at exactly zero mean relative drift. Reading the values out settles why: across 266,240 elements `ckpt_259` has *one* distinct value, 1.0,

Table 3: Sampled-tier corpus rows. Columns as in Table 1.

Family	Stage pair	D	S	Ratio	Artefact
Tulu-3-8B	SFT→DPO	2	1	6.17×–6.49×	a7-dense-chains-sampled
Tulu-3.1-8B	DPO→final	3	1	4.30×–4.35×	a8-crosslab-sampled
zephyr-7b-beta	SFT→DPO	2	1	3.06×–3.09×	a8-crosslab-sampled
OLMo-2-13B	DPO→Instruct	3	1	5.17×–6.33×	a8-crosslab-sampled
OLMoE-0125	DPO→Instruct	3	1	5.38×–5.41×	a9-moe-sampled
JetMoE-8B	SFT→Chat	2	1	9.69×–17.82×	a9-moe-sampled
DeepSeek-R1-Zero	base→R1-Zero	1	1	4.94×	a5-codec-bakeoff
Qwen3-30B-A3B	base→post-tr.	1	1	1.72×	a5-codec-bakeoff
gpt-oss-20b (126 children)	selection survey	1–2	1	survey	g5-selection-survey

Table 4: Amber, anchor ckpt_200, whole-model, BF16 both sides. Checkpoints past 259 are published at FP32 and are excluded; the controls that establish this are in § H.1.

Signed d	≈tokens	Ratio	Mean rel. drift	Mean cosine
+1	3.5B	2.794×	2.945×10^{-2}	0.99933
−1	3.5B	2.788×	2.983×10^{-2}	0.99931
+2	7B	2.688×	3.954×10^{-2}	0.99879
+4	14B	2.602×	5.200×10^{-2}	0.99791
+8	28B	2.535×	6.650×10^{-2}	0.99658
+16	56B	2.487×	8.162×10^{-2}	0.99485
+32	112B	2.458×	9.423×10^{-2}	0.99315
+59	207B	2.451×	1.003×10^{-1}	0.99224
−64	224B	1.676×	1.900×10^{-1}	0.97260
−128	448B	1.489×	3.818×10^{-1}	0.89906
−200	700B	1.389×	1.327	0.46987

Table 5: Amber format controls. $B/param$ is delta bytes over Amber’s nominal 6.7B parameters; the divisor cancels in the $4.05 \times$.

Pair	d	dtypes	Ratio	$B/param$	Drift
200→201	1	bf16→bf16	2.794×	0.720	2.945×10^{-2}
259→260	1	bf16→fp32	1.381×	2.914	2.993×10^{-2}
260→261	1	fp32→fp32	1.379×	2.917	3.068×10^{-2}
136→200	64	bf16→bf16	1.676×	1.200	1.900×10^{-1}
264→328	64	fp32→fp32	1.229×	3.274	2.327×10^{-1}

which is the initialisation, while ckpt_260 has 119,082 (g2b-norm-probe-a259-a260.json). The first 260 of Amber’s 360 published checkpoints do not carry their trained norms. For the ratios this changes nothing — they measure the artefacts as published, which is what a storage paper should measure, and 65 norm tensors are 0.004 % of the model — but the family’s whole-tensor identity rate is an artefact of the export and this paper quotes none.

I Pre-registration

Every prediction scored in the body was registered by commit before the run that tested it, and the commit is quoted at the point of use. The registrations are 8b70b32 (codec bake-off), 3c7e816 (row granularity), d32051b (parallel writes), 8147380 (the Tulu-3-70B RLVR whole-model diff) and ccccd306 (the eight wave-1 gaps: the Tulu-3-70B SFT→DPO diff, the

OLMo-2 ladder, the Amber ladder, the selection survey, the serving breadth, the verification A/B, the integer-difference lane, and the lossless baselines). Verdicts, including the misses, are in the observation or evaluation subsection each belongs to; none is reported only here.