Access Sets Matter: Budgeting Expert Reads for Scalable Weight-Space Model Merging

Yuanyi Wang [1] **Yanggan Gu** [1] **Su Lu** [1] **Yifan Yang** [1]
Zhaoyi Yan [2] **Congkai Xie** [2] **Jianmin Wu** [1] **Hongxia Yang** [1 2 3]

Code: wyy-code/mergepipe

Abstract

Weight-space model merging is usually formulated as an algebraic operation on checkpoints, yet at LLM scale the limiting resource is often the set of expert weights that must be read. We introduce **MergePipe**, a budget-aware execution layer that casts LLM merging as an *expert access-set* problem: given a merge operator and a checkpoint family in a shared weight coordinate system, choose which expert delta blocks to access under an explicit I/O budget. MergePipe indexes parameter blocks, builds deterministic access plans, and executes the induced budgeted merge with replayable manifests. The plan is budget-sound by construction and recovers the full-read merge at full budget; for fixed-coefficient additive operators, the omitted-update error is bounded by the norm of omitted deltas. Across Qwen and Llama merging workloads, MergePipe reduces expert-read I/O by up to an order of magnitude and achieves up to $11\times$ speedups. Representative budget sweeps show $O(10^{-3})$ parameter deviation from full-read merges and no monotonic degradation on downstream benchmarks.

1. Introduction

Modern LLM development increasingly produces *checkpoint families*: a shared base model, instruction-tuned variants, domain experts, and adapter or delta updates (Wortsman et al., 2022; Yadav et al., 2023). These families are becoming weight-space datasets, and model merging offers a post-training primitive for consolidating them into one deployable model without ensembling or another full training run (Yang et al., 2024; Lu et al., 2024; Wang et al., 2025).

Most merging work asks how task vectors, signs, sparsity masks, or low-rank updates should be combined. For LLM-scale checkpoint families, an equally important execution question appears: *which expert parameters must be read to realize a merge?* Naive merge scripts treat checkpoints as opaque files, scan all expert parameters, apply AVG (Wortsman et al., 2022), TIES (Yadav et al., 2023), DARE-like rules (Yu et al., 2024), and write the output. As the expert pool grows, this expert-read term scales nearly linearly in the number of experts K, making iterative merging I/O-bound rather than compute-bound.

MergePipe assumes the standard homologous setting where experts are fine-tuned from a shared base, or have already been aligned into a common weight coordinate system. It is therefore complementary to symmetry- or permutation-alignment methods: after weights live in one coordinate chart, MergePipe asks which expert deltas should be accessed under a finite budget.

Our thesis is that LLM-scale merging needs a *weight-access abstraction*, not only better merge rules. MergePipe treats expert deltas as a budgeted resource and decouples *which* blocks are accessed from *how* accessed deltas are combined. Full-read merging fixes access mask $A = \mathbf{1}$; budgeted merging chooses an access mask A under cost $C_{\text{expert}}(A) \leq B$ and executes the induced mask-aware operator Ψ_{op}. Omitted entries are encoded by the mask and do not trigger storage reads. Thus, full-budget execution recovers the standard merge, while lower budgets define an explicit approximation and expose a speed–fidelity frontier.

Our contributions are as follows: **(i)** We introduce expert access sets as a budgeted object for weight-space model merging. **(ii)** We prove budget soundness, full-budget consistency, and an omitted-update bound for additive merges. **(iii)** We instantiate this abstraction in MergePipe, achieving up to order-of-magnitude expert-I/O reduction and $11\times$ speedups on Qwen and Llama checkpoint families, with rep-

Corresponding author: Hongxia Yang. [1]The Hong Kong Polytechnic University, PolyU [2]InfiX.ai [3]Hong Kong Polytechnic University Daya Bay Technology and Innovation Research Institute. Correspondence to: Yuanyi Wang <wangyuanyi713@gmail.com>, Hongxia Yang <hongxia.yang@polyu.edu.hk>.

Accepted at the ICML 2026 Workshop on Weight-Space Symmetries: from Foundations to Practical Applications.

resentative budget sweeps preserving downstream behavior.

2. Budgeted Access Sets

Consider merging a base model M_0 with experts $\{M_i\}_{i=1}^K$. Let $\mathcal{T}$ be the tensor set and $\mathcal{B}_t$ the deterministic blocks of tensor t. For block (t, b), let $M_0[t, b]$ be the base block and $\Delta_{i,t,b} = M_i[t, b] - M_0[t, b]$ expert i's delta. A full-read merge computes

$$M_{\text{full}}[t, b] = M_0[t, b] + \Phi_{\text{op},t,b}(\Delta_{1,t,b}, \ldots, \Delta_{K,t,b}; \omega)\,, \tag{1}$$

where Φ_{op} is AVG, TIES, DARE, or another weight-space operator; ω denotes the mask for randomized operators.

MergePipe exposes the hidden execution choice with an access mask

$$A \in \{0, 1\}^{|\mathcal{U}|}, \quad A_{i,t,b} = 1 \Leftrightarrow \text{read expert } i \text{ for } (t, b), \tag{2}$$

where $\mathcal{U} = \{(i, t, b) : i \in [K], t \in \mathcal{T}, b \in \mathcal{B}_t\}$. Each unit has physical read cost $c_{i,t,b} \geq 0$, measured in the executor's accounting unit. The controllable expert-read cost is

$$C_{\text{expert}}(A) = \sum_{(i,t,b)\in\mathcal{U}} A_{i,t,b} c_{i,t,b}, \tag{3}$$

while base reads and output writes are checkpoint-boundary costs. The budgeted merge is

$$M_A[t, b] = M_0[t, b] + \Psi_{\text{op},t,b}\left(A_{\cdot,t,b}, \Delta_{\cdot,t,b}; \omega\right), \tag{4}$$

with full-budget consistency

$$\Psi_{\text{op},t,b}(\mathbf{1}, \Delta_{\cdot,t,b}; \omega) = \Phi_{\text{op},t,b}(\Delta_{1,t,b}, \ldots, \Delta_{K,t,b}; \omega). \tag{5}$$

We require Ψ_{op} to be *non-anticipatory*: if two delta tuples agree on all selected entries $\{i : A_{i,t,b} = 1\}$, they produce the same budgeted output. Hence omitted entries are represented only by the mask, not by reading their contents. Offline ANALYZE reads used to build sketches or norms are amortized catalog construction; if performed inside a merge run, they are counted in $C^{\text{run}}_{\text{expert}}$.

Planning can be written as the idealized access-set objective

$$A^\star = \arg\max_A \sum_{i,t,b} A_{i,t,b} s(i, t, b) \quad \text{s.t.} \quad C_{\text{expert}}(A) \leq B, \tag{6}$$

where $s(i, t, b)$ comes from norms, sketches, coverage, or fallback metadata. MergePipe implements this objective with deterministic greedy or score-per-byte heuristics rather than exact knapsack optimization.

Proposition 1 (Budgeted execution invariant)**.** *Let A be an access mask with $C_{\text{expert}}(A) \leq B$. Assume the planner and executor use the same nonnegative read costs $c_{i,t,b}$, and the charged execution trace reads only selected expert blocks, i.e., $N^{\text{run}}_{i,t,b} \leq A_{i,t,b}$. Then*

$$C^{\text{run}}_{\text{expert}}(A) = \sum_{i,t,b} N^{\text{run}}_{i,t,b} c_{i,t,b} \leq C_{\text{expert}}(A) \leq B. \tag{7}$$

Moreover, if $A = \mathbf{1}$, then the budgeted operator recovers the full-read merge, $M_A = M_{\text{full}}$; for randomized operators, this equality is pathwise under the same seed or mask ω.

Corollary 1 (Expert-read fraction under fixed absolute budget)**.** *If full-read merging reads all K expert checkpoints with average expert-read cost $\bar{C} > 0$, then $C^{\text{full}}_{\text{expert}}(K) = K\bar{C}$. Under a fixed absolute expert-read budget B,*

$$\frac{C^{\text{run}}_{\text{expert}}(A)}{C^{\text{full}}_{\text{expert}}(K)} \leq \frac{B}{K\bar{C}}. \tag{8}$$

Thus, when $\bar{C}$ is bounded below and B does not scale with K, the expert-read fraction decreases as $O(1/K)$.

Proposition 2 (Additive omission bound)**.** *Assume blocks form a disjoint partition of the parameter vector. Suppose the budgeted additive operator uses fixed coefficients independent of A:*

$$\Psi_{\text{add},t,b}(A_{\cdot,t,b}, \Delta_{\cdot,t,b}) = \sum_{i=1}^{K} \alpha_{i,t,b} A_{i,t,b} \Delta_{i,t,b}. \tag{9}$$

Then

$$M_{\text{full}}[t, b] - M_A[t, b] = \sum_{i=1}^{K} (1 - A_{i,t,b}) \alpha_{i,t,b} \Delta_{i,t,b}. \tag{10}$$

Let $q_{t,b}(A) = \sum_i (1 - A_{i,t,b}) |\alpha_{i,t,b}| \|\Delta_{i,t,b}\|_2$. Then

$$\|M_{\text{full}} - M_A\|_2 \leq \left[\sum_{t,b} q_{t,b}(A)^2\right]^{1/2} \leq \sum_{t,b} q_{t,b}(A). \tag{11}$$

where M_{full} denotes the same additive operator with $A = 1$.

Proposition 2 gives a simple justification for norm- or sketch-based access scores in fixed-coefficient additive merges. With $u_{i,t,b} = |\alpha_{i,t,b}| \|\Delta_{i,t,b}\|_2$, minimizing the looser ℓ_1 omission bound is equivalent to maximizing retained utility $\sum_{i,t,b} A_{i,t,b} u_{i,t,b}$ under the expert-read budget. This fidelity bound does not apply to selected-only renormalization, TIES, or DARE, where access masks can change coefficients, random drops, sparsification, or sign election; for these operators, MergePipe provides the same budgeted execution abstraction, while fidelity is evaluated empirically.

Remark: Proofs are provided in Appendix A.3.

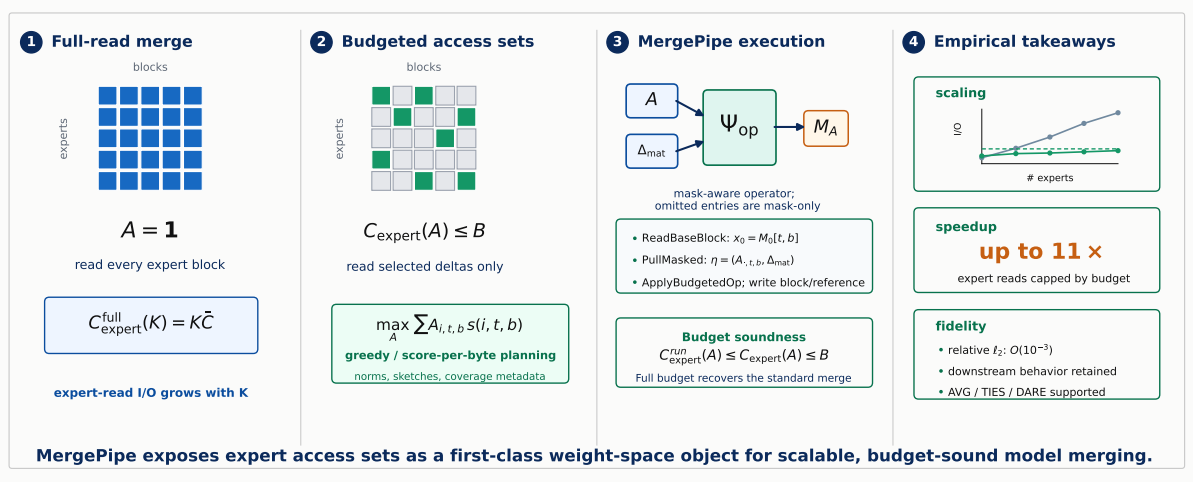

Figure 1. **Budgeted access sets in weight space.** Full-read merging fixes $A = \mathbf{1}$. MergePipe chooses a budget-feasible access mask A and executes the induced mask-aware operator Ψ_{op}; omitted entries are represented by the mask and do not trigger expert reads.

3. MergePipe

MergePipe realizes the access-mask abstraction through a catalog–plan–execute loop; detailed algorithms are in Appendix A.2. Given a base checkpoint, expert pool, merge operator, and expert-I/O budget, it returns a logical merged checkpoint and a replayable manifest recording the access mask, plan hash, touched blocks, realized expert reads, and lineage. Figure 1 illustrates how budgeted access sets are planned and executed in weight space. More details of the system are shown in Figure 4.

Catalog. MergePipe exposes checkpoints as block-structured weight data rather than opaque files. For each tensor block, the catalog stores size, layout, hashes, and lightweight statistics such as sketches or coverage hints. These reusable metadata allow cost estimation and access planning without repeatedly scanning all expert checkpoints.

Planner. The planner constructs an access mask A satisfying $C_{\text{expert}}(A) \leq B$. It ranks candidate expert deltas using catalog statistics and selects blocks under the expert-I/O budget, with deterministic tensor-level fallback when block metadata are missing. The planner does not introduce a new merge rule: it decides which deltas are physically materialized, while the requested operator is applied through its mask-aware instantiation Ψ_{op}, recovering the standard full-read operator at $A = \mathbf{1}$.

Executor. The executor streams base blocks in checkpoint order and uses `DeltaIterator` to materialize only selected expert deltas from full checkpoints, explicit deltas, or LoRA-style adapters (Hu et al., 2022). Omitted entries are passed as mask values rather than storage reads. For each block, MergePipe applies Ψ_{op} and records touched blocks, contributing experts, and realized I/O in the manifest. Our comparisons use full logical checkpoint materialization and isolate expert-read I/O before optional overlay optimization.

Algorithm 1 PLANGEN: greedy budget-aware plan

Require: Experts $\{M_i\}_{i=1}^{K}$, catalog $\mathcal{C}$, operator op, budget B
Ensure: Budget-feasible merge plan π

Build candidate expert-block set $\mathcal{Q} = \{(i, t, b)\}$ from block metadata.
Score candidates $s(i, t, b)$ using norms, sketches, coverage, or deterministic fallback metadata.
Sort $\mathcal{Q}$ by decreasing score with stable tensor/block tie-breaking.
$\mathcal{R}_\pi \leftarrow \emptyset$, $\widehat{C} \leftarrow 0$.
for candidate $(i, t, b) \in \mathcal{Q}$ **do**
  if $\widehat{C} + c_{i,t,b} \leq B$ **then**
    $\mathcal{R}_\pi \leftarrow \mathcal{R}_\pi \cup \{(i, t, b)\}$; $\widehat{C} \leftarrow \widehat{C} + c_{i,t,b}$.
  end if
end for
Record operator parameters, traversal order, selected-block digest, and $\widehat{C}$.
return $\pi = (\text{op}, \theta, \mathcal{R}_\pi, \text{order})$.

4. Experiments

We evaluate the causal chain behind MergePipe: full-read expert access grows with K; budgeted access caps this growth; wall time follows expert-read I/O; and the resulting approximation remains useful downstream. Experiments cover Qwen3-0.6B/1.7B/8B, Llama-3.2-3B, and Llama-3.1-8B, with up to 20 experts for Qwen/Llama-8B and 25 for Llama-3.2-3B. Merges are CPU-only, use SSD-backed storage, and disable OS-level file caching unless stated otherwise. Baselines implement the same operators but scan all required expert checkpoints on every invocation. Appendix B reports the budget sweep, I/O and overhead breakdowns, a compact operator table, and the full quality table.

Scaling and budget behavior. Figure 2 shows that naive

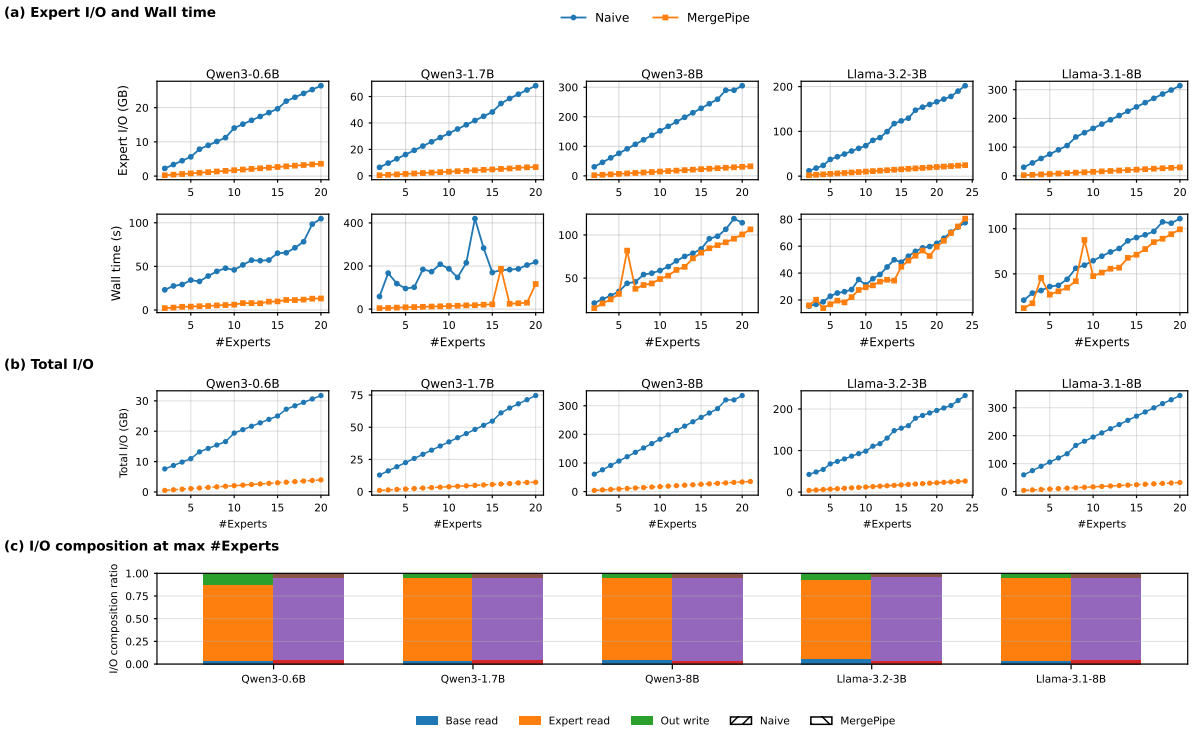

Figure 2. **Scaling with the number of experts.** Full-read merging repeatedly scans expert checkpoints, so expert-read I/O and wall time grow with K. MergePipe enforces a fixed expert-I/O budget, keeping expert reads nearly flat and shifting the remaining cost toward the unavoidable checkpoint boundary.

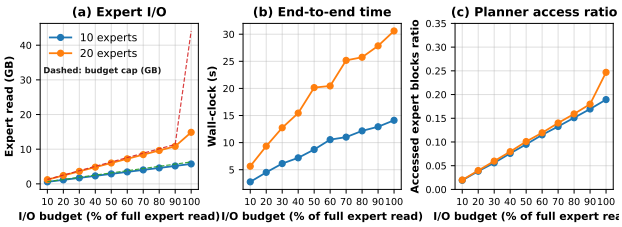

Figure 3. **Budget-aware planning behavior.** **(a)** Realized expert reads grow monotonically with the requested I/O budget and remain under the cap. **(b)** End-to-end wall time follows expert-read volume. **(c)** The fraction of accessed expert blocks expands smoothly as more budget is allocated.

merging has near-linear expert-I/O growth and matching wall-time growth as K increases. MergePipe keeps the access set within B, making expert reads nearly flat and yielding order-of-magnitude expert-I/O reductions and up to $11\times$ speedups. Budget sweeps from 10% to 100% further show monotone realized expert-read I/O, accessed-block ratio, and wall-time growth (Appendix B), making B a direct throughput–fidelity knob.

Operator generality. On Llama-3.1-8B, MergePipe reduces expert-read I/O across AVG, TIES, and DARE because access planning precedes local merge semantics. Sparse methods benefit most: at $K{=}8$, TIES I/O drops from 79.5GB to 3.46GB and wall time from 614s to 51s; at $K{=}20$, TIES still reads 3.46GB while the naive pipeline exceeds 174GB, with a 70.4% wall-time reduction.

Table 1. **Operator generality on Llama-3.1-8B.** MergePipe applies the same access-budget abstraction to AVG, TIES, and DARE. I/O is reported in GB.

Op.	K	Naive I/O	MP I/O	Naive s / MP s	Reduc.
AVG	8	70.7	3.46	136.2 / 64.1	53.0%
AVG	20	171.0	3.46	363.0 / 338.4	6.8%
TIES	8	79.5	3.46	614.3 / 51.1	91.7%
TIES	20	174.4	3.46	1203.7 / 356.3	70.4%
DARE	8	79.5	3.46	99.1 / 44.7	54.9%
DARE	20	173.2	3.46	464.2 / 414.4	10.7%

Table 2. **Fidelity under budgeted expert access.** Qwen3-0.6B, TIES, $K{=}20$. Budget is normalized to the full-read TIES endpoint; touched ratio is measured after TIES sparsification.

Budget	Touched	Rel. ℓ_2	HumanEval	IFEval	DROP
1.0	0.247	0	39.71	68.56	66.81
0.9	0.180	7.23×10^{-4}	42.68	68.82	67.23
0.7	0.140	8.30×10^{-4}	37.80	68.47	66.26
0.5	0.101	8.84×10^{-4}	40.24	68.11	66.00

Quality under bounded access. Budgeted access is a controlled approximation, so we measure both parameter deviation and downstream quality. Table 2 compares budgeted TIES outputs with the full-read output on Qwen3-0.6B with 20 experts. The full-read TIES endpoint has touched ratio below one because TIES itself sparsifies updates; the budget is normalized to this operator-induced access cost. Even at 0.5 budget, relative ℓ_2 deviation remains $O(10^{-3})$.

Algorithm 2 EXECUTEMERGE: budget-enforced streaming execution

Require: Plan π, base M_0, experts $\{M_i\}$, storage $\mathcal{S}$, catalog $\mathcal{C}$, transaction manager $\mathcal{T}$
Ensure: Snapshot id sid and manifest man
$\mathcal{T}$.BEGIN(); open staging writer w; initialize touch and coverage maps.
for tensor t in π.TENSORORDER() **do**
  Initialize `DeltaIterator` D for t.
  for block b in π.BLOCKSTOMATERIALIZE(t) **do**
    Read base block $x_0 \leftarrow \mathcal{S}$.READBASEBLOCK($M_0, t, b$).
    $\eta \leftarrow D$.PULLMASKED($b, \mathcal{R}_\pi$).
    $x \leftarrow$ APPLYBUDGETEDOP(x_0, η, π.OP(t, b)).
    w.WRITEBLOCKORREFERENCE(t, b, x, M_0); update touch and coverage.
  end for
end for
Validate hashes; build man $\leftarrow \mathcal{C}$.BUILDMANIFEST(π, *touch*, *coverage*).
$sid \leftarrow \mathcal{T}$.ATOMICPUBLISH($w$, man); $\mathcal{C}$.COMMITRECORD(sid, man).
$\mathcal{T}$.COMMIT(); **return** (sid, man).

Table 3. **Parameter deviation and downstream quality under different budgets** (Qwen3-0.6B, TIES, K=20).

	Deviation from full-read merge			Downstream score		
Budget	Touched	Rel. ℓ_2	P95 block	HumanEval	IFEval	DROP
1.0	0.247	0	0	39.71	68.56	66.81
0.9	0.180	7.23×10^{-4}	3.66×10^{-3}	42.68	68.82	67.23
0.8	0.159	7.77×10^{-4}	3.66×10^{-3}	36.59	68.94	67.08
0.7	0.140	8.30×10^{-4}	3.98×10^{-3}	37.80	68.47	66.26
0.6	0.119	8.30×10^{-4}	3.98×10^{-3}	37.37	67.87	66.10
0.5	0.101	8.84×10^{-4}	3.98×10^{-3}	40.24	68.11	66.00

HumanEval (Chen, 2021), IFEval (Zhou et al., 2023), and DROP (Dua et al., 2019) stay close to the full-read baseline and show no monotonic degradation, indicating a favorable speed–fidelity frontier in this setting.

System overhead and scope. Planning and metadata are small compared with tensor streaming: in a Qwen3-0.6B, 16-expert run, planning takes 1.21s (about 1% of execution), the manifest is 812KB, and catalog storage is 3.79% of total I/O (Appendix B). MergePipe targets offline, iterative merging with many disk-resident experts; gains naturally shrink for small expert sets, dense full-read regimes, or GPU-resident in-memory fusion.

5. Related Work

Model merging. Weight-space merging methods combine checkpoints or task vectors into one deployable model. Recent surveys summarize the broader model-fusion and model-merging landscape (Yang et al., 2024; Lu et al., 2024; Zhou et al., 2025; 2026). Dense methods include model soups (Wortsman et al., 2022); sparse or interference-aware methods include TIES, DARE, low-rank variants, and activation- or sensitivity-informed merging (Yu et al., 2024; Liu et al., 2025b; Nobari et al., 2025; Liu et al., 2025a). Recent LLM work further studies post-merge feature calibration and quantization (Gu et al., 2026; Wang et al., 2026b), continual post-training conflicts (Wang et al., 2026e), and domain-specific expert composition in weight space (Wang et al., 2026a). Together, these works study *how* weights should be combined, adapted, or deployed. MergePipe is complementary: it studies *which expert weights must be accessed* when checkpoints and expert pools are large.

Table 4. Execution and system costs (Qwen3-0.6B, 16 experts).

Metric	Absolute value	Relative cost
Planning time	1.21s	1.04% of execution
Execution time	116.62s	100%
Executed expert-read I/O	23,349MB	1.18× pre-plan
Metadata catalog size	3,468MB	3.79% of total I/O
Execution manifest size	812KB	<0.001%

Machine Learning Management Systems. Large-scale LLM development produces many checkpoints, deltas, and merged variants, motivating systems for experiment tracking, artifact logging, versioning, workflow orchestration, and provenance (Zaharia et al., 2018; Barreto Simedo Pacheco et al., 2024; Vadde & Munagandla, 2024; Eggers, 2024; Bux et al., 2015; Green et al., 2007; Ruan et al., 2021). These systems improve reproducibility and traceability, but they mainly manage checkpoints and pipelines rather than optimizing the parameter I/O pattern of LLM merging. MergePipe addresses this bottleneck by treating parameters as block-level execution units, caching reusable tensor statistics, and planning merge execution under an explicit expert-read budget. This enables predictable, budget-aware checkpoint access for large-scale LLM merging, complementing prior ML management systems.

6. Conclusion

We presented MergePipe, a budgeted access-set abstraction for scalable weight-space model merging. Our central observation is that, for LLM checkpoint families, merging is constrained not only by how expert deltas are combined, but also by which expert deltas are physically read. By making expert access a first-class budgeted resource, MergePipe separates the logical merge rule from its physical access pattern, provides budget-sound and full-budget-consistent execution, and exposes a practical speed–fidelity frontier. Across Qwen and Llama checkpoint families, this view turns full-read expert scans into bounded expert access, yielding large I/O and runtime reductions while retaining downstream behavior in representative budgeted merges. More broadly, our results suggest that as model families continue to grow, weight-space methods should be paired with execution layers that treat model weights as structured, budgeted data rather than opaque checkpoint files.

Acknowledgements

This paper is fully supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. T41-517/25-N).

References

Barreto Simedo Pacheco, L., Rahman, M., Rabbi, F., Fathollahzadeh, P., Abdellatif, A., Shihab, E., Chen, T.-H., Yang, J., and Zou, Y. Dvc in open source ml-development: The action and the reaction. In *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI*, pp. 75–80, 2024.

Bux, M., Brandt, J., Lipka, C., Hakimzadeh, K., Dowling, J., and Leser, U. Saasfee: scalable scientific workflow execution engine. *Proceedings of the VLDB Endowment*, 8(12):1892–1895, 2015.

Chen, M. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Dua, D., Wang, Y., Dasigi, P., Stanovsky, G., Singh, S., and Gardner, M. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2368–2378, 2019.

Eggers, S. Automating data lineage and pipeline extraction. *Proceedings of the VLDB Endowment. ISSN*, 2150:8097, 2024.

Green, T. J., Karvounarakis, G., and Tannen, V. Provenance semirings. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 31–40, 2007.

Gu, Y., Wang, Y., Yan, Z., Zhang, Y., Zhou, Q., Wu, F., and Yang, H. Infifpo: Implicit model fusion via preference optimization in large language models. *arXiv preprint arXiv:2505.13878*, 2025.

Gu, Y., Cai, S., Wang, Z., Wang, W., Wang, Y., Wang, P., Huang, S., Lu, S., Wu, J., and Yang, H. Featcal: Feature calibration for post-merging models. *arXiv preprint arXiv:2605.13030*, 2026.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Liu, S., Wu, H., He, B., Han, X., Yuan, M., and Song, L. Sens-merging: Sensitivity-guided parameter balancing for merging large language models. *arXiv preprint arXiv:2502.12420*, 2025a.

Liu, Z., Wu, H., Yao, Y., She, R., Han, X., Zhong, T., and Yuan, M. Lore-merging: Exploring low-rank estimation for large language model merging. *arXiv preprint arXiv:2502.10749*, 2025b.

Lu, J., Pang, Z., Xiao, M., Zhu, Y., Xia, R., and Zhang, J. Merge, ensemble, and cooperate! a survey on collaborative strategies in the era of large language models. *arXiv preprint arXiv:2407.06089*, 2024.

Nobari, A. H., Alim, K., ArjomandBigdeli, A., Srivastava, A., Ahmed, F., and Azizan, N. Activation-informed merging of large language models. *arXiv preprint arXiv:2502.02421*, 2025.

Ruan, P., Dinh, T. T. A., Lin, Q., Zhang, M., Chen, G., and Ooi, B. C. Lineagechain: a fine-grained, secure and efficient data provenance system for blockchains. *The VLDB Journal*, 30(1):3–24, 2021.

Vadde, B. C. and Munagandla, V. Devops in the age of machine learning: Bridging the gap between development and data science. *International Journal of Machine Learning Research in Cybersecurity and Artificial Intelligence*, 15(1):530–544, 2024.

Wang, P., Liu, P., Wang, Y., Chen, G., Ren, X., Li, X., Hao, Z., Kong, Y., Zhang, Q., and Ni, D. Discovering physical directions in weight space: Composing neural pde experts. *arXiv preprint arXiv:2605.14546*, 2026a.

Wang, W., Gu, Y., Cai, S., Wang, Y., Wang, P., Wu, J., and Yang, H. E-pmq: Expert-guided post-merge quantization with merged-weight anchoring. *arXiv preprint arXiv:2605.16882*, 2026b.

Wang, Y., Gu, Y., Zhang, Y., Zhou, Q., Yan, Z., Xie, C., Wang, X., Yuan, J., and Yang, H. Model merging scaling laws in large language models. *arXiv preprint arXiv:2509.24244*, 2025.

Wang, Y., Lu, S., Gu, Y., Wang, P., Yang, Y., Yan, Z., Xie, C., Wu, J., and Yang, H. Not all disagreement is learnable: Token teachability in on-policy distillation, 2026c.

Wang, Y., Yan, Z., Zhang, Y., Zhou, Q., Gu, Y., Wu, F., and Yang, H. Infigfusion: Graph-on-logits distillation via efficient gromov-wasserstein for model fusion. *Advances in Neural Information Processing Systems*, 38:119677–119713, 2026d.

Wang, Y., Yang, Y., Lu, S., Gu, Y., Wang, P., Wang, W., Yan, Z., Xie, C., Wu, J., Cao, J., et al. Geometry conflict: Explaining and controlling forgetting in llm continual post-training. *arXiv preprint arXiv:2605.09608*, 2026e.

Wortsman, M., Ilharco, G., Gadre, S. Y., Roelofs, R., Gontijo-Lopes, R., Morcos, A. S., Namkoong, H.,

Farhadi, A., Carmon, Y., Kornblith, S., et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pp. 23965–23998. PMLR, 2022.

Yadav, P., Tam, D., Choshen, L., Raffel, C. A., and Bansal, M. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36:7093–7115, 2023.

Yang, E., Shen, L., Guo, G., Wang, X., Cao, X., Zhang, J., and Tao, D. Model merging in llms, mllms, and beyond: Methods, theories, applications and opportunities. *arXiv preprint arXiv:2408.07666*, 2024.

Yu, L., Yu, B., Yu, H., Huang, F., and Li, Y. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Forty-first International Conference on Machine Learning*, 2024.

Zaharia, M., Chen, A., Davidson, A., Ghodsi, A., Hong, S. A., Konwinski, A., Murching, S., Nykodym, T., Ogilvie, P., Parkhe, M., et al. Accelerating the machine learning lifecycle with mlflow. *IEEE Data Eng. Bull.*, 41 (4):39–45, 2018.

Zhou, J., Lu, T., Mishra, S., Brahma, S., Basu, S., Luan, Y., Zhou, D., and Hou, L. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

Zhou, Q., Zhang, Y., Gu, Y., Wang, Y., Sang, Z., Yan, Z., Li, Z., Zhang, S., Wu, F., and Yang, H. Democratizing ai through model fusion: A comprehensive review and future directions. *Nexus*, 2025.

Zhou, Q., Zhang, Y., Gu, Y., Wang, Y., Yan, Z., Li, Z., Chung, C. Y., and Yang, H. Model fusion for scalable and sustainable artificial intelligence: A review and outlook. *Journal of Modern Power Systems and Clean Energy*, 14 (1):37–49, 2026.

A. Formalization and Algorithms

This section expands the formal model, merge-operator semantics, catalog schema, and execution algorithms used by MergePipe.

Figure 4 gives the implementation view behind the access-set abstraction in the main text. MergePipe indexes LLM checkpoints as block-level weight data, plans budget-feasible expert-delta access, executes the requested mask-aware merge operator, and publishes a logical checkpoint together with a replayable manifest. This system view is complementary to the main formulation: the central object remains the expert access mask, while the runtime components make that mask executable at checkpoint scale.

A.1. Cost Accounting and Operator Semantics

The total merge cost decomposes into base reads, expert reads, output writes, and metadata:

$$C_{\text{merge}} = C_{\text{base}} + C_{\text{expert}} + C_{\text{out}} + C_{\text{meta}}. \tag{12}$$

MergePipe constrains only the expert-read term. For a sparse plan $\mathcal{R}_\pi$, the induced access mask is

$$\begin{aligned} A_{i,t,b} &= \mathbf{1}\{(i,t,b) \in \mathcal{R}_\pi\}, \\ \widehat{C}_{\text{expert}}(\pi) &= \sum_{(i,t,b)\in\mathcal{R}_\pi} c_{i,t,b} \le B, \end{aligned} \tag{13}$$

where $c_{i,t,b}$ is the same physical accounting unit used by the executor. In uncompressed block-aligned runs, it equals the stored byte length of block b.

For fixed-coefficient additive operators, the mask can be implemented through a zero-completed tuple

$$\bar{\Delta}^A_{i,t,b} = A_{i,t,b}\Delta_{i,t,b}. \tag{14}$$

The zero entries are logical handles and do not trigger storage reads. Average merging is then

$$\Psi_{\text{AVG}}(A,\Delta) = \sum_{i=1}^{K} \alpha_i \bar{\Delta}^A_i, \qquad \sum_i \alpha_i = 1. \tag{15}$$

DARE applies the same idea before random drop/rescale, using a fixed seed or mask ω when compared to the full-read endpoint. TIES trims, elects signs, and averages sign-consistent entries after masking. Because access masks can alter random drops, top-k sparsification, and sign election, we do not claim a global smooth error bound for TIES/DARE; instead we report parameter deviation and downstream quality empirically.

If an implementation renormalizes an additive merge over only selected experts, the omitted-delta bound in the main text no longer applies. Let the full-read additive merge

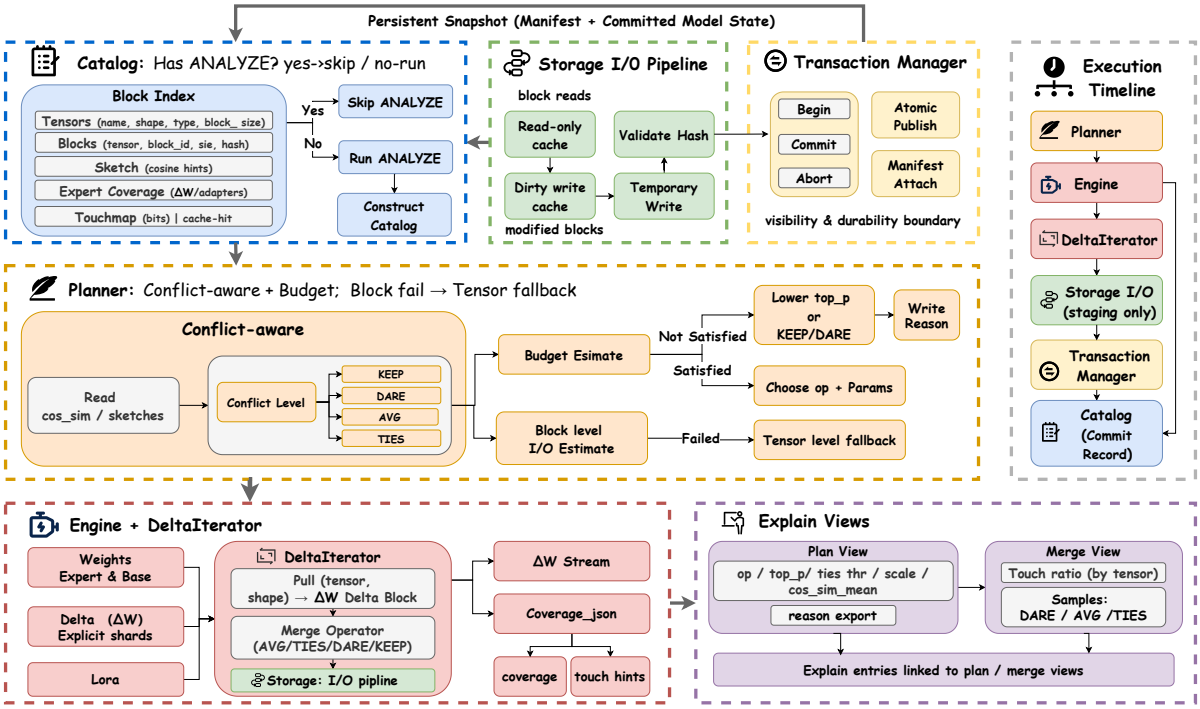

Figure 4. **MergePipe system overview.** The runtime realizes budget-aware weight-space merging through block-level cataloging, access-set planning, mask-aware execution, and manifest-based replay. The planner controls expert-delta reads under the I/O budget, while the executor streams only selected expert blocks and materializes the resulting logical checkpoint.

use coefficients $\alpha_{i,t,b}$, and let the selected-only budgeted merge use coefficients $\beta_{i,t,b}(A)$ with $\beta_{i,t,b}(A) = 0$ whenever $A_{i,t,b} = 0$. Then

$$M_{\text{full}}[t,b] - M_A[t,b] = \sum_i (\alpha_{i,t,b} - \beta_{i,t,b}(A))\Delta_{i,t,b}, \quad (16)$$

and, defining

$$r_{t,b}(A) = \sum_i |\alpha_{i,t,b} - \beta_{i,t,b}(A)| \|\Delta_{i,t,b}\|_2, \quad (17)$$

we have

$$\|M_{\text{full}} - M_A\|_2 \le \left[\sum_{t,b} r_{t,b}(A)^2 \right]^{1/2}. \quad (18)$$

This coefficient-drift form is the correct bound for selected-only averaging.

A.2. Planning and Execution Sketches

The planning and execution sketches are included in the main text as Algorithms 1 and 2.

A.3. Proofs

Proof of Proposition 1. For every expert block (i,t,b), assumption (ii) gives $N^{\text{run}}_{i,t,b} \le A_{i,t,b}$. Since $c_{i,t,b} \ge 0$,

$$\begin{aligned} C^{\text{run}}_{\text{expert}}(A) &= \sum_{i,t,b} N^{\text{run}}_{i,t,b} c_{i,t,b} \le \sum_{i,t,b} A_{i,t,b} c_{i,t,b} \\ &= C_{\text{expert}}(A) \le B. \end{aligned} \quad (19)$$

This proves budget soundness. For full-budget consistency, suppose $B \ge C_{\text{expert}}(\mathbf{1})$ and the planner selects $A = \mathbf{1}$. By Equation (5), for every block (t,b),

$$\begin{aligned} M_A[t,b] &= M_0[t,b] + \Psi_{\text{op},t,b}(\mathbf{1}, \Delta_{\cdot,t,b}; \theta, \omega) \\ &= M_0[t,b] + \Phi_{\text{op},t,b}(\Delta_{\cdot,t,b}; \theta, \omega) = M_{\text{full}}[t,b]. \end{aligned}$$

Thus $M_A = M_{\text{full}}$ blockwise. For randomized operators, the equality is pathwise under the same seed or mask ω; without fixing ω, it is equality in distribution. □

Proof of Corollary 1. By Proposition 1, $C^{\text{run}}_{\text{expert}}(A) \le B$. Full-read execution reads every expert and incurs $C^{\text{full}}_{\text{expert}}(K) = K\bar{C}$. Dividing by $K\bar{C}$ gives the claim. □

Proof of Proposition 2. For the full-read and budgeted ad-

ditive merges,

$$M_{\text{full}}[t,b] = M_0[t,b] + \sum_{i=1}^{K} \alpha_{i,t,b}\Delta_{i,t,b},$$

$$M_A[t,b] = M_0[t,b] + \sum_{i=1}^{K} \alpha_{i,t,b}A_{i,t,b}\Delta_{i,t,b}.$$

Subtracting gives

$$M_{\text{full}}[t,b] - M_A[t,b] = \sum_{i=1}^{K}(1 - A_{i,t,b})\alpha_{i,t,b}\Delta_{i,t,b}.$$

Let $d_{t,b} = M_{\text{full}}[t,b] - M_A[t,b]$. Since blocks are disjoint parameter coordinates,

$$\|M_{\text{full}} - M_A\|_2 = \left(\sum_{t,b} \|d_{t,b}\|_2^2\right)^{1/2}.$$

The triangle inequality gives $\|d_{t,b}\|_2 \leq q_{t,b}(A)$. Substitution yields the first inequality in Equation (11); the second follows from $(\sum z_{t,b}^2)^{1/2} \leq \sum z_{t,b}$ for nonnegative $z_{t,b}$. □

Atomic visibility follows from Algorithm 2: a run either publishes one snapshot/manifest pair or leaves no externally visible state.

B. Additional Experimental Evidence

This supplement preserves the non-duplicated additional evidence most directly used by the main text: budget-controlled planning behavior and I/O/overhead decomposition. Baselines implement the same merge operator as MergePipe, use full-read expert access, and run with OS-level file caching disabled.

Budget-Controlled Planning. The budget-controlled planning results are included in the main text as Figure 3.

I/O Breakdown and Overhead. Figure 5 shows that the gains come from reducing the expert-read term rather than from metadata effects. Base reads and output writes are checkpoint-boundary costs, while planning and transactional overhead remain small compared with tensor streaming.

C. Limitations

MergePipe targets budgeted weight-space access for checkpoint merging and is complementary to behavior-level fusion methods based on preference optimization, on-policy distillation, or logit-space alignment (Gu et al., 2025; Wang et al., 2026c;d). It assumes experts share a common weight

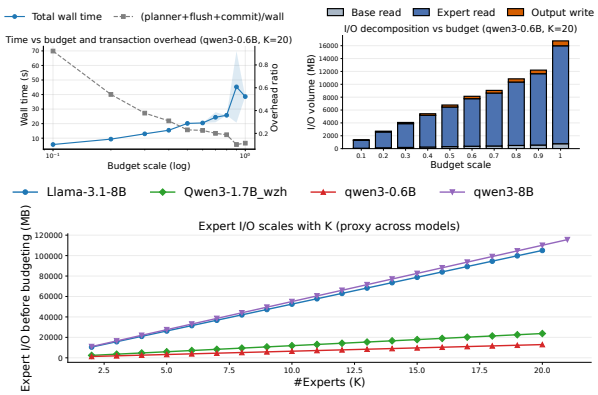

Figure 5. **Where MergePipe saves time. Top-left:** planning, flush, and commit are small relative to execution. **Top-right:** tightening the budget primarily removes expert reads, while base reads and output writes remain nearly fixed. **Bottom:** before budgeting, expert reads scale with the number of experts.

coordinate system and does not address permutation, symmetry, or representation alignment. Budgeted merging is approximate: only the full-budget setting recovers the standard full-read merge, while lower budgets rely on mask-aware execution. Its quality is therefore empirical, especially for nonlinear sparse operators such as TIES and DARE, and the benefits are largest when disk-resident expert-read I/O dominates.