MECEFO-V2: EFFICIENT FAULT-TOLERANT OPTIMIZATION FOR MOE TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Sparse Mixture-of-Experts (MoE) models have become a central architecture for scaling large language models, but they also reshape the fault-tolerance problem during distributed training. Compared with dense models, MoE training couples hardware failures with expert parallelism, routing-dependent workload imbalance, and substantially enlarged training state, making conventional failover strategies inefficient. Existing MoE fault-tolerant methods primarily optimize checkpointing or elastic recovery, but leave open whether the training computation itself can be adapted online after failures with bounded memory and compute overhead. In this work, we propose MeCeFO-v2, a fault-tolerant optimization framework for MoE training under hybrid pipeline, expert, and data parallelism. Instead of assigning a failed worker's workload to a single pipeline neighbor, MeCeFO-v2 redistributes it cooperatively across expert-parallel peers, improving expert update balance and reducing the approximation burden on each surrogate worker. To keep failover training within resource budget, MeCeFO-v2 introduces two structure-aware approximations: norm-guided selective head retention for attention layers and router-aware sequence-level low-rank compression for MoE experts. We further develop an online controller that automatically selects the attention-head budget and expert compression ratio by minimizing profiled gradient error subject to memory and compute constraints. Theoretically, we establish the same $\mathcal{O}(1/\sqrt{nT})$ convergence rate as standard distributed SGD under bounded gradient error. Experiments on large-scale MoE training demonstrate that MeCeFO-v2 maintains validation perplexity and downstream performance close to fault-free training while limiting throughput degradation to 0.72–6.15%, compared with 7.61–57.92% for prior MoE fault-tolerance baselines.

1 INTRODUCTION

Mixture-of-Experts (MoE) architectures have emerged as a key path for scaling large language models (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022; Du et al., 2022; Rajbhandari et al., 2022; Liu et al., 2024a;b). Recent frontier-scale models increasingly rely on sparse routing and expert parallelism to improve training efficiency at scale (Fedus et al., 2022; Gale et al., 2023; Dai et al., 2024; Liu et al., 2024a;b). Under this paradigm, each token is routed to only a small subset of experts, creating a fundamentally different training state structure compared to dense transformers: the model state is sharded not only across layers and data samples, but also across specialized expert modules that are activated dynamically based on the input.

As MoE training scales to large clusters, hardware failures become unavoidable and often occur with alarming frequency. For instance, Alibaba reports a downtime percentage of 31.19% for handling failures (Dong et al., 2024), and Meta notes that hardware issues trigger a failure every 4 hours on average during LLaMA-3-405B training (Grattafiori et al., 2024). The frequency of failures tends to increase with cluster size (Dean & Barroso, 2013), making fault tolerance critical for training efficiency and GPU utilization. Critically, a single hardware failure in MoE training does not merely remove a generic compute shard; it can simultaneously disrupt expert availability, perturb the routing-dependent load balance, and inflate recovery overhead due to the large and distributed expert state.

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

Existing fault-tolerant training methods have largely followed system-centric designs such as checkpointing, elastic rescheduling, or redundant execution (Mohan et al., 2021; Thorpe et al., 2023; Jang et al., 2023; Wang et al., 2024; Wu et al., 2023; Lian et al., 2025; Maurya et al., 2024). Recent work has adapted these mechanisms to MoE training: MoC-System reduces checkpointing overhead through partial-expert checkpointing (Cai et al., 2025), MoEvement further introduces sparse-to-dense reconstruction for efficient recovery (Gandhi & Kozyrakis, 2026), and Lazarus explores expert replication to bypass rollback (Wu et al., 2024). While effective, these methods remain grounded in preserving or recovering the original training computation, which can introduce significant resource and efficiency overhead.

A different perspective is introduced by **MeCeFO** (Hu et al., 2025), which argues that strict fidelity to the original training trajectory is unnecessarily restrictive. Indeed, the ultimate goal of training is not to reproduce exact intermediate computations, but to obtain model parameters that generalize well. MeCeFO leverages this insight by continuing training *online* under failures with memory- and computation-efficient approximations, achieving substantially higher throughput than prior methods that are based on exact state recovery. However, MeCeFO was designed for dense transformer models under conventional data and pipeline parallelism. Extending its online failover philosophy to MoE training raises a natural but challenging open question:

Can we extend this approximate optimization principle to MoE models and their expert-parallel training paradigm, while still delivering both fault tolerance and high training throughput?

Answering this question for MoE models is substantially harder than for dense transformers. MoE training involves pipeline, expert, and data parallelism (Fedus et al., 2022; Rajbhandari et al., 2022), so a failure simultaneously disrupts layer placement, expert sharding, and token dispatch. Naively extending MeCeFO's neighbor-do-both strategy creates a severe expert-update imbalance: approximated and full-rank gradients mix across experts, breaking optimization symmetry. Moreover, importance in MoE models is inherently heterogeneous across attention heads and routed tokens, making uniform compression policies suboptimal. Finally, the optimal approximation configuration should depend on the *live* failure pattern, routing distribution, and available memory and compute.

In this work, we propose **MeCeFO-v2**, a robust fault-tolerant optimization framework for sparse MoE training under hybrid PP-EP-DP parallelism. Its core is cooperative failover: when a device fails, the failed computation is redistributed across all healthy expert-parallel peers in an EP-aware manner, which naturally balances load and spreads the failover burden evenly.

Building on this failover mechanism, we design memory- and compute-efficient on-failure training strategies tailored to MoE structures and EP parallelism. (i) For attention layers, we retain only the κ heads with the largest value-output norms for backward computation and discard the rest, saving memory while preserving informative gradients. (ii) For MoE layers, we apply low-rank weight-gradient compression selectively to token–expert assignments: within each sequence, the assignments with the largest routing scores stay full-rank, and the remaining ones are compressed. This sequence-level, router-aware decision respects expert importance and avoids token-wise instability. (iii) An online controller adaptively selects head budgets and compression ratios under resource constraints, yielding a favorable efficiency–performance trade-off across varying failure and load conditions. We further prove that MeCeFO-v2 maintains the $\mathcal{O}(1/\sqrt{nT})$ convergence rate of distributed SGD under bounded gradient error.

Our contributions are three-fold:

- We propose **MeCeFO-v2**, a fault-tolerant optimization framework for sparse MoE training that integrates EP-cooperative failover with structured approximation strategies.
- We establish convergence guarantees and empirically validate the effectiveness of our attention and expert approximation policies.
- We experimentally show that MeCeFO-v2 outperforms prior MoE fault-tolerance baselines, MoC-System (Cai et al., 2025), MoEvement (Gandhi & Kozyrakis, 2026), and Lazarus (Wu et al., 2024), in fault-time efficiency: prior baselines suffer 7.61–57.92% throughput drops under failures, whereas MeCeFO-v2 limits throughput degradation to 0.72–6.15%, with comparable model performance.

2 Related Works

Fault-tolerant training for MoE models. Existing fault-tolerant training methods have largely followed system-centric designs such as checkpointing or elastic rescheduling (Mohan et al., 2021; Thorpe et al., 2023; Jang et al., 2023; Wang et al., 2024; Wu et al., 2023; Lian et al., 2025; Maurya et al., 2024). Recent MoE-oriented systems redesign fault tolerance around sparse expert state and locality-aware recovery. Representative approaches include partial-expert checkpointing (Cai et al., 2025), sparse checkpointing with sparse-to-dense reconstruction (Gandhi & Kozyrakis, 2026), dynamic checkpoint transfer scheduling (Xie et al., 2025), and expert replication (Wu et al., 2024). These methods improve recovery efficiency, but they still aim to reconstruct or preserve the original training computation. MeCeFO relaxes this requirement by continuing training online under failures with memory- and computation-efficient approximations (Hu et al., 2025). Our MeCeFO-v2 extends this online failover paradigm to hybrid DP-PP-EP MoE training: it distributes failover across EP peers and applies structured approximations to the resulting extra expert workload.

Selective attention head computation. Foundational studies have shown that many attention heads are redundant and can be removed with little performance loss (Michel et al., 2019; Voita et al., 2019). Building on this observation, a line of prior work studies attention-head importance estimation and selective head pruning or computation, spanning meta-pruning, differentiable subset selection, dynamic inference-time pruning, and entropy-aware pruning (Zhang et al., 2021; Guo & Sheridan, 2026; Li et al., 2021; Jaradat et al., 2024; Choi et al., 2025; Sandoval-Segura et al., 2025). Among these works, value vectors and head-output norms have been shown to provide more reliable signals than attention weights for identifying unimportant heads (Sandoval-Segura et al., 2025). Beyond pruning-oriented methods, mechanistic analyses of active-dormant heads and attention sink phenomena provide complementary evidence that head utility is highly non-uniform and context-dependent (Guo et al., 2024a; Gu et al., 2025; Xiao et al., 2024; Sandoval-Segura et al., 2025; Fu et al., 2026). MeCeFO-v2 builds on this observation and selects important heads based on value-output norms, rather than static pruning or the MHA-skipping strategy used in MeCeFO.

MoE compression and dynamic sparsity. Existing MoE and FFN compression methods exploit heterogeneous importance in feed-forward layers and sparse expert activation (Cheng et al., 2025; Wu et al., 2025; Cai et al., 2024; Tan et al., 2025). These approaches reduce MoE cost through finer-grained sparsity, expert refactoring, decoupling, or reuse. However, they change the routing structure or deployed expert modules, making them less suitable for our failover setting, where the base MoE execution should remain unchanged and compression is only introduced under temporary failure-induced resource pressure. Dynamic sparse routing, i.e., allowing expert budgets to vary across tokens (Jin et al., 2025; Wen et al., 2025), is more closely related to our compression strategy. The key idea is to allocate fewer experts to easier inputs while assigning more capacity to harder ones. We adopt this resource-allocation view, but do not introduce a new routing policy: instead, MeCeFO-v2 leverages routing logits and activation statistics as importance signals for router-aware, sequence-level low-rank compression.

3 Preliminaries

3.1 MoE Transformer Model

We consider a standard Transformer with mixture-of-experts (MoE) layers. For simplicity of exposition, we describe the attention module using conventional multi-head attention (MHA) notation. For layer ℓ, let $\mathbf{X}^{(\ell-1)} \in \mathbb{R}^{n\times d}$ denote the input hidden states. For a MHA layer with H_ℓ attention heads, the output of each head $h \in [H_\ell]$ and the MHA layer is

$$\mathbf{O}_{\ell,h} = \mathrm{Softmax}\Big(\mathbf{Q}_{\ell,h}\mathbf{K}_{\ell,h}^\top/\sqrt{d}\Big)\,\mathbf{V}_{\ell,h}, \quad \mathrm{MHA}_\ell(\mathbf{X}^{(\ell-1)}) = \big[\mathbf{O}_{\ell,1};\ldots;\mathbf{O}_{\ell,H_\ell}\big]W_\ell^O,$$

where $\mathbf{Q}_{\ell,h}, \mathbf{K}_{\ell,h}, \mathbf{V}_{\ell,h}$ are the query, key, and value representations of head h, respectively.

For a MoE layer with M_ℓ experts $\{E_{\ell,i}\}_{i=1}^{M_\ell}$, each expert $E_{\ell,j} : \mathbb{R}^d \to \mathbb{R}^d$ is typically a two-layer MLP. For each token $\mathbf{x}$, a router with weight $\mathbf{W}_{\ell,g}$ computes $\mathbf{z}_\ell(\mathbf{x}) = \mathrm{Softmax}(\mathbf{W}_{\ell,g}\mathbf{x}) \in \mathbb{R}^{M_\ell}$ and selects top-K_ℓ experts $\mathcal{A}_\ell(\mathbf{x})$ with normalized weights $\pi_{\ell,j}(\mathbf{x}) = \mathbf{z}_\ell(\mathbf{x})_j / \sum_{i\in\mathcal{A}_\ell(\mathbf{x})} \mathbf{z}_\ell(\mathbf{x})_i$. The MoE output for token $\mathbf{x}$ is $\sum_{j\in\mathcal{A}_\ell(\mathbf{x})} \pi_{\ell,j}(\mathbf{x})\, E_{\ell,j}(\mathbf{x})$.

In the following, we focus on a single layer ℓ and omit the layer index when no ambiguity arises.

3.2 Distributed Parallelism and Fault Model

We consider hybrid DP-PP-EP training with degrees (D, P, E), where a rank is indexed by $r = (d, p, e)$. We use $\mathbf{w}$ to denote model parameters and $f(\mathbf{w})$ to denote training objective. For an MoE layer on pipeline stage p, experts are sharded across EP ranks as a disjoint partition $\{\Omega_e\}_{e=1}^{E}$, where Ω_e is owned by EP rank e.

We model a failure event as a set F of failed ranks, covering both single-GPU and node-level failures; we present the single-rank case $F = \{r_f\}$ with $r_f = (d_f, p_f, e_f)$ and extend it to general F in Sec. 4.1. Since DP maintains replicas of the same model shard, the failed rank's parameters and optimizer states can be fetched from an alive replica. The central problem is how to execute the failed workload online without exceeding the original stage memory and compute budgets.

3.3 Review of MeCeFO

MeCeFO (Hu et al., 2025) is a fault-tolerant training method for dense Transformer models under hybrid DP+PP parallelism. Upon failure, a neighboring rank takes over computation while missing states are restored from DP replicas. MeCeFO further introduces module-specific approximations: (1) for MHA modules, it **skips** the MHA connections and avoids storing attention activations and computing their gradients in the spirit of DropBP (Woo et al., 2024); (2) for FFN modules, it adopts activation recomputation. It further reduces the FFN cost by approximating the weight gradient in a low-rank subspace using leading singular vectors.

4 Method

MeCeFO-v2 is centered on EP-cooperative failover: after a failure, the missing MoE capacity is recovered within the affected EP group by redistributing the failed expert workload across surviving ranks, as shown in Figure 1. To keep this recovery within memory and compute budgets, we introduce structured approximations in forward and backward computation (Figure 2), including head-aware attention retention and router-aware MoE activation compression.

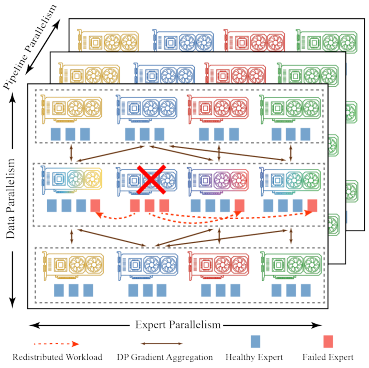


Figure 1: Overview of the MeCeFO-v2 framework.

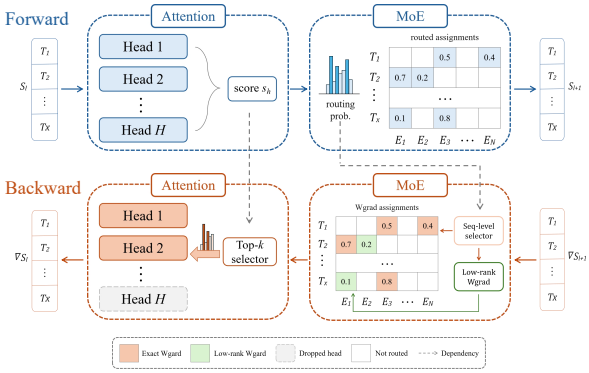
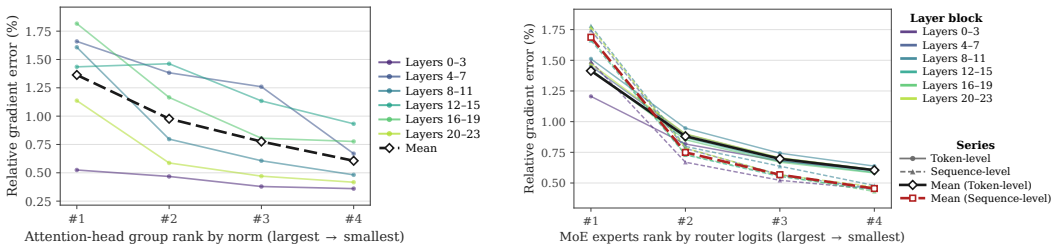


Figure 2: Forward and backward computation in MeCeFO-v2.

4.1 EP-Cooperative Failover

Unlike MeCeFO-v1, which hands a failed shard to a single PP neighbor and thus concentrates both load and approximation on one expert subset, MeCeFO-v2 performs failover within the affected EP group. When a rank $r_f = (d_f, p_f, e_f)$ fails, let Ω_{e_f} denote the set of experts originally hosted by the failed EP rank e_f and $\mathcal{S}^{-f}_{d_f,p_f} := \{(d_f, p_f, e) : e \in [E] \setminus \{e_f\}\}$ be the surviving EP ranks in the same DP-PP stage. All ranks in $\mathcal{S}^{-f}_{d_f,p_f}$ enter a common protection mode as *protected ranks*



(a) Attention heads (b) MoE assignments

Figure 3: Relative gradient error versus importance scores on Qwen3-MoE 16B-A2B. (Left) Gradient error is positively correlated with head importance measured by value-output norm. (Right) Gradient error is positively correlated with routing scores; within a token, ranking by router logits is equivalent to ranking by routing scores. These results justify retaining high-importance heads and high-confidence expert assignments.

and cooperatively absorb the failed expert workload using a balanced assignment map $\varphi_f : \Omega_{e_f} \to [E] \setminus \{e_f\}$ such that $\left|\left|\varphi_f^{-1}(e)\right| - |\Omega_{e_f}|/(E-1)\right| \le 1$ for all $e \in [E] \setminus \{e_f\}$. Each surviving rank $e \neq e_f$ is assigned the subset $\Omega_{e_f \to e} := \varphi_f^{-1}(e)$, which is only about $1/(E-1)$ of the missing shard. The same construction applies to a general failure set F, e.g., all ranks on a failed node: for each affected EP group, we take the union of the experts hosted by its failed ranks and distribute it evenly across its surviving EP peers on other nodes, while the rest of MeCeFO-v2 is unchanged (Appendix D.1).

After the redistribution, we will use the techniques proposed in the following Sections 4.2 and 4.3 to derive the gradient $\widetilde{\mathbf{G}}_d^{(t)}$ of each DP replica d at step t (exact for replicas without failed ranks). The DP interface remains unchanged, i.e., $\overline{\mathbf{G}}^{(t)} = \frac{1}{D}\sum_{d=1}^{D} \widetilde{\mathbf{G}}_d^{(t)}$.

4.2 Head-Aware Partial Attention Retention

Next we introduce the first approximation knob for the protected ranks, which targets attention activations. In MeCeFO-v1, the MHA connections are simply skipped, which discards all head-level gradients. This may introduce large deviations in training dynamics as some heads are essential for network representation (Michel et al., 2019).

MeCeFO-v2 instead applies a finer-grained strategy by retaining a subset of important attention heads. Inspired by the value-aware pruning metric proposed in Guo et al. (2024b), for a given attention layer and budget $\kappa \in [H]$, MeCeFO-v2 computes the forward value-output energy and retains the top-κ heads

$$s_h := \|\mathbf{O}_h\|_F^2, \quad \mathbf{O}_h := \mathrm{Softmax}\Big(\mathbf{Q}_h\mathbf{K}_h^\top/\sqrt{d}\Big)\,\mathbf{V}_h, \quad \mathcal{H}^\star(\kappa) := \mathrm{TopK}(\{s_h\}_{h=1}^{H}, \kappa).$$

Intuitively, selecting the top-κ heads minimizes the residual energy of the truncated representation and yields a close approximation to the original MHA output. Under the common L-smoothness condition, i.e., $\|\nabla f(\mathbf{O}) - \nabla f(\mathbf{O}')\|_2 \le L\|\mathbf{O} - \mathbf{O}'\|_2$, this approximation in turn bounds the induced gradient error and helps preserve the fidelity of the backward signal. We validate this choice empirically in Fig. 3 (left): discarding heads with smaller value-output norms leads to lower gradient error.

In implementation, the forward computation remains exact for all heads. Only the backward cache is changed: head-local activations are retained for $h \in \mathcal{H}^\star(\kappa)$ and discarded otherwise. Thus larger κ gives higher gradient fidelity, while smaller κ saves more activation memory and backward computation. The resource accounting for this knob is combined with the MoE knob in Sec. 4.4.

4.3 Router-Aware Sequence-Level Expert Compression

The second approximation knob targets the extra MoE workload introduced by EP-cooperative failover. After a failed expert shard is redistributed across the surviving EP ranks, MeCeFO-v2 keeps

the router and expert forward computation exact, but approximates the weight gradient (Wgrad) of a subset of token-expert assignments in a low-rank subspace; the input gradient (Dgrad) of all assignments and the Wgrad of the remaining ones are computed exactly.

Given a compression budget χ, measured as the average number of compressed token–expert assignments per token, MeCeFO-v2 performs sequence-level selection rather than enforcing a fixed quota for each token. For a local sequence q with token set $\mathcal{T}_q$ and the corresponding expert assignment set $\mathcal{R}_q := \{(\mathbf{x}, j) : \mathbf{x} \in \mathcal{T}_q,\ j \in \mathcal{A}(\mathbf{x})\}$, the compressed assignment set $\mathcal{C}_q(\chi)$ is picked through

$$\mathcal{C}_q(\chi) \in \arg\min_{\mathcal{C} \subseteq \mathcal{R}_q} \sum_{(\mathbf{x},j)\in\mathcal{C}} \mathbf{z}(\mathbf{x})_j, \quad \text{s.t. } |\mathcal{C}| = \chi|\mathcal{T}_q|,$$

with $\chi|\mathcal{T}_q|$ rounded to an integer. Equivalently, this compresses the assignments with the smallest routing scores $\mathbf{z}(\mathbf{x})_j$ and keeps the most confident ones at full rank. As shown in Fig. 3 (right), assignments with lower routing scores incur substantially smaller gradient error when compressed, supporting our sequence-level selection strategy. This sequence-level rule naturally adapts to uneven token difficulty: easy tokens may undergo more aggressive compression, while hard tokens retain more full-rank expert updates.

For each compressed assignment, consider one expert linear sublayer $\mathbf{Y} = \mathbf{X}\mathbf{W}^\top$. The exact gradient w.r.t. $\mathbf{W}$ is $\mathbf{G}_W = \mathbf{\Delta}^\top \mathbf{X}$ with $\mathbf{\Delta} = \partial\mathcal{L}/\partial\mathbf{Y}$. MeCeFO-v2 applies a low-rank approximation based on singular value decomposition (SVD). To reduce the cost of SVD, MeCeFO-v2 updates the low-rank basis every T_{SVD} steps using a recently profiled gradient matrix collected from a held-out microbatch. Let $\mathbf{V}_{j,r}$ denote the top-r right singular vectors obtained from truncated SVD, which provides the best rank-r approximation. The projected activation and the corresponding approximate Wgrad is then given by

$$\widehat{\mathbf{X}} := \mathbf{X}\mathbf{V}_{j,r}, \qquad \widetilde{\mathbf{G}}_W = \mathbf{\Delta}^\top \widehat{\mathbf{X}} \mathbf{V}_{j,r}^\top.$$

The routing score directly controls the resulting error. Since the output of expert j is scaled by $\pi_j(\mathbf{x})$, we have $\mathbf{\Delta} = \mathbf{\Pi}_j \bar{\mathbf{\Delta}}$ with $\mathbf{\Pi}_j := \mathrm{diag}(\pi_j(\mathbf{x}))$ and a gate-free upstream gradient $\bar{\mathbf{\Delta}}$. Restricting $\bar{\mathbf{\Delta}}$ and $\mathbf{X}$ to the compressed assignments of expert j gives

$$\|\mathbf{G}_W - \widetilde{\mathbf{G}}_W\|_F \le \pi_j \, \|\bar{\mathbf{\Delta}}\|_F \, \|\mathbf{X}(\mathbf{I} - \mathbf{V}_{j,r}\mathbf{V}_{j,r}^\top)\|_2, \qquad \pi_j := \max\{\pi_j(\mathbf{x}) : (\mathbf{x}, j) \text{ is compressed}\},$$

so compressing low-score assignments keeps the error bound small (derivation in Appendix B.4).

4.4 Budget-Aware Automatic Parameter Selection

The two knobs $\theta = (\kappa, \chi)$ are selected per failure episode rather than fixed globally, since the feasible region depends on EP degree, routing distribution, sequence length, microbatch size, and available memory headroom. At protection time, MeCeFO-v2 profiles memory and compute to form a feasible set

$$\Theta_p := \left\{ \theta : \mathrm{Mem}_p(\theta) \le M_{\max}, \quad \mathrm{Comp}_p(\theta) \le \mathrm{Comp}_p^{\mathrm{base}} \right\},$$

where Mem_p and Comp_p denote the per-rank memory and computation of stage p under protection (see Appendix B.3), $M_{\max}$ is the device memory capacity, and $\mathrm{Comp}_p^{\mathrm{base}}$ is the per-rank computation of stage p in fault-free training. MeCeFO-v2 aims to minimize the estimated gradient error. Specifically, every T_{prof} steps during protection mode, the runtime profiles a small candidate set on a held-out microbatch. Let $\mathbf{g}_{\mathrm{ref}}^{\mathrm{prof}}$ be the fault-free reference gradient and $\widetilde{\mathbf{g}}^{\mathrm{prof}}(\theta)$ be the protected gradient under policy θ. We estimate the relative gradient error $\varepsilon(\theta)$ and fit it using a quadratic surrogate of κ and χ as follows:

$$\varepsilon(\theta) = \|\widetilde{\mathbf{g}}^{\mathrm{prof}}(\theta) - \mathbf{g}_{\mathrm{ref}}^{\mathrm{prof}}\|_2 / \|\mathbf{g}_{\mathrm{ref}}^{\mathrm{prof}}\|_2, \quad \widehat{\varepsilon}(\kappa, \chi) := c_0 + c_1\kappa + c_2\chi + c_3\kappa^2 + c_4\chi^2 + c_5\kappa\chi.$$

The final policy is selected as $\theta^\star := \arg\min_{\theta\in\Theta_p} \widehat{\varepsilon}(\theta)$. Detailed resource accounting is presented in Appendix B.

4.5 Overall Training Workflow and Convergence

Algorithm 1 summarizes the full training-time behavior; the protected forward and backward passes are given in Algorithms 2 and 3 in Appendix C.

Algorithm 1 MeCeFO-v2 training framework

Input: Initial model weights and optimizer states; DP/PP/EP degrees (D, P, E); T_{prof}.
Output: Final model weights.

Initialize all ranks as standard ranks and all redistribution maps $\varphi_{\ell,F}$ as empty.
For each training step $t = 0, \ldots, T-1$**:**
  For each rank (d, p, e) **in parallel:**
    Detect failure or recovery events in the local PP-EP neighborhood.
    Failure of rank set F**:**
      Fetch parameters and optimizer states from an alive DP replica.
      Build balanced maps $\varphi_{\ell,F}$ and assign the failed expert workload.
      Profile if needed and select $\theta^\star = (\kappa, \chi)$.
    Recovery of rank set F **(at an iteration boundary):**
      Fetch the latest parameters and full Adam states from an alive DP replica.
      Restore the original expert placement, reset $\varphi_{\ell,F}$, and exit protection mode.
    Every T_{prof} **steps during protection:**
      Periodically refresh gradient-subspace bases, update $\widehat{\varepsilon}$, and reselect $\theta^\star$ over Θ_p.
    Per layer:
      Execute `MeCeFOv2_Forward`$(\ell, \theta^\star)$.
      Execute `MeCeFOv2_Backward`$(\ell, \theta^\star)$ with head retention and low-rank Wgrad.
      Average DP gradients and update full model states.

These algorithms expose two invariants that are important for correctness. First, failover changes expert ownership within the EP group but does not change the DP averaging rule seen by the optimizer. Second, protection mode preserves the forward computation and all full-size model states; approximation affects only activation caching and the backward computations that depend on those cached activations.

Finally, we prove the convergence of MeCeFO-v2 in a nonconvex setting using momentum SGD. A simplified version is stated below; the formal statement and proof are in Appendix A.

Theorem 1 (Rate of MeCeFO-v2, informal)**.** *Under L-smoothness, bounded gradient variance, and bounded relative gradient error (Assumptions 2–4), with suitable learning rate η and momentum β_1,*

$$\frac{1}{T+1}\sum_{t=0}^{T}\mathbb{E}\left[\|\nabla f(\boldsymbol{w}^{(t)})\|_2^2\right] = \mathcal{O}\left(1/\sqrt{nT}\right),$$

where n is the number of samples per iteration and $\mathcal{O}(\cdot)$ hides constants and lower-order terms.

5 Experimental Results

5.1 Experimental Setup

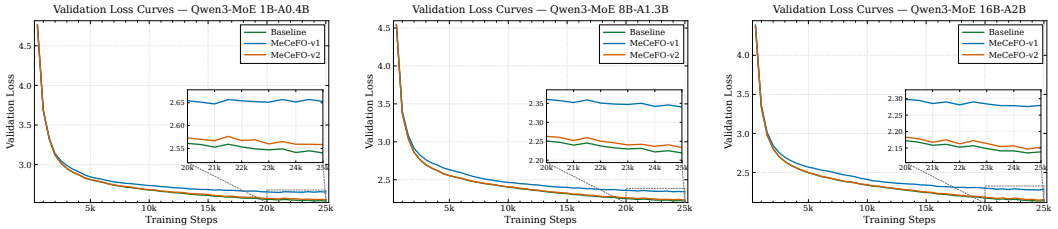
Cluster setup. We conducted experiments on a 128-GPU cluster composed of 16 nodes, each with eight NVIDIA H20 GPUs. Intra-node communication leveraged NVLink (900 GB/s), and inter-node communication used InfiniBand (400 Gb/s).

Models and training. We evaluate three Qwen3-MoE-style models (Yang et al., 2025), Qwen3-MoE 1B-A0.4B, 8B-A1.3B, and 16B-A2B. All models are trained on FineWeb-Edu (Penedo et al., 2024) for 50B tokens with sequence length 1024 and global batch size 2048 using a Megatron-based (Shoeybi et al., 2019) codebase. Unless otherwise specified, training uses hybrid parallelism with $\mathrm{DP} = 8$, $\mathrm{EP} = 4$, and $\mathrm{PP} = 4$. Additional model architecture and training details are provided in Appendix B.1 and Appendix B.2.

Failure Scenario. We simulate five controlled failure modes by varying the average failure duration and recovery duration, measured in hours. Table 1 summarizes the settings and assigns each mode a short name used in the following experiments. The first sweep fixes the recovery time at 4 hours and

Table 2: Validation PPL under different failure scenarios. Lower is better.

Model	No Fault	Sparse-Fail	Base-Fail	Frequent-Fail	Slow-Recover	Fast-Recover
Qwen3-MoE 1B-A0.4B	12.68	12.72	12.83	12.90	12.92	12.75
Qwen3-MoE 8B-A1.3B	9.20	9.26	9.29	9.33	9.32	9.24
Qwen3-MoE 16B-A2B	8.48	8.53	8.54	8.59	8.57	8.50



(a) Qwen3-MoE 1B-A0.4B (b) Qwen3-MoE 8B-A1.3B (c) Qwen3-MoE 16B-A2B

Figure 4: Validation loss curves across Qwen3-MoE model scales under the Frequent-Fail scenario.

changes the average failure time from 2 to 1 and 0.5 hours, capturing increasingly frequent failures. The second sweep fixes the average failure time at 1 hour and varies the recovery time from 2 to 4 and 8 hours, capturing increasingly long recovery periods.

Throughput measurement. All throughput is measured end to end over the entire failure trace, including failure detection, communication reconfiguration, state transfer, and warm-up; all methods share the same failure detector and accounting (Appendix C).

5.2 Convergence Experiments

Table 2 reports the validation perplexity (PPL) of MeCeFO-v2 under the five failure scenarios in Table 1. Across all model scales, the PPL remains close to the no-failure baseline despite changes in either failure frequency or recovery duration. For the Qwen3-MoE 1B-A0.4B model, the largest increase is only 0.24 PPL under Slow-Recover; for the Qwen3-MoE 8B-A1.3B and Qwen3-MoE 16B-A2B models, the largest increases are 0.13 and 0.11 PPL, respectively. These small gaps indicate that MeCeFO-v2 preserves convergence quality even when failures become more frequent or recovery takes longer.

Scenario	(Fail, Recover time)	Sweep role
Sparse-Fail	(2.0, 4.0)	Fail sweep
Base-Fail	(1.0, 4.0)	Shared baseline
Frequent-Fail	(0.5, 4.0)	Fail sweep
Slow-Recover	(1.0, 8.0)	Recover sweep
Fast-Recover	(1.0, 2.0)	Recover sweep

Table 1: Failure scenarios used in our experiments. Failure time and recovery time are average durations in hours.

Figure 4 further compares the validation loss trajectories across the three model scales. MeCeFO-v2 closely tracks the no-failure curve, showing that its protection-mode approximations introduce only minor optimization noise and do not significantly affect the learned model capability. By contrast, MeCeFO-v1 causes visibly larger loss deviations in the MoE setting because its coarse failover approximation is less aligned with expert-parallel sparse execution.

5.3 Throughput Experiments

Table 3 reports training throughput (token/s) under different failure scenarios. MeCeFO-v2 keeps throughput close to its no-failure baseline across all model scales and failure settings. For the Qwen3-MoE 1B-A0.4B, Qwen3-MoE 8B-A1.3B, and Qwen3-MoE 16B-A2B models, MeCeFO-v2 limits throughput degradation to 0.72–4.38%, 2.49–5.28%, and 2.97–6.15%, respectively, relative to its no-failure baseline. In contrast, the other fault-tolerance baselines suffer substantial throughput drops, especially under frequent failures and slow recovery: MoC-System loses more than 40% throughput in several Qwen3-MoE 8B-A1.3B and Qwen3-MoE 16B-A2B cases, MoEvement exceeds 40% loss on Qwen3-MoE 16B-A2B under frequent failures and slow recovery, and MeCeFO-v1 and Lazarus

Table 3: Training throughput (token/s) under different failure scenarios. "k" is the suffix for thousands. Percentages in parentheses denote the relative change from each method's no-failure baseline.

Model	Method	No Fault	Sparse-Fail	Base-Fail	Frequent-Fail	Slow-Recover	Fast-Recover
Qwen3-MoE 1B-A0.4B	MeCeFO-v1	508.51k	469.79k (-7.61%)	460.45k (-9.45%)	439.32k (-13.61%)	443.81k (-12.72%)	464.11k (-8.73%)
	MoC-System	465.94k	393.68k (-15.51%)	353.47k (-24.14%)	283.19k (-39.22%)	298.03k (-36.04%)	372.31k (-20.09%)
	Lazarus	459.21k	406.38k (-11.50%)	392.35k (-14.56%)	369.49k (-19.54%)	384.09k (-16.36%)	401.91k (-12.48%)
	MoEvement	476.22k	411.59k (-13.57%)	374.49k (-21.36%)	294.83k (-38.09%)	298.27k (-37.37%)	403.34k (-15.30%)
	MeCeFO-v2	**583.65k**	**579.43k (-0.72%)**	**574.56k (-1.56%)**	**558.07k (-4.38%)**	**559.90k (-4.07%)**	**571.48k (-2.09%)**
Qwen3-MoE 8B-A1.3B	MeCeFO-v1	246.96k	226.37k (-8.34%)	219.94k (-10.94%)	212.82k (-13.82%)	214.36k (-13.20%)	221.35k (-10.37%)
	MoC-System	214.60k	172.56k (-19.59%)	149.07k (-30.54%)	119.71k (-44.22%)	127.23k (-40.71%)	162.49k (-24.28%)
	Lazarus	212.33k	182.52k (-14.04%)	171.19k (-19.38%)	161.98k (-23.71%)	163.95k (-22.79%)	177.18k (-16.55%)
	MoEvement	219.23k	181.38k (-17.26%)	163.09k (-25.61%)	131.94k (-39.82%)	135.37k (-38.25%)	176.43k (-19.52%)
	MeCeFO-v2	**271.68k**	**263.65k (-2.96%)**	**264.91k (-2.49%)**	**257.34k (-5.28%)**	**259.06k (-4.65%)**	**261.91k (-3.60%)**
Qwen3-MoE 16B-A2B	MeCeFO-v1	184.29k	165.79k (-10.04%)	161.19k (-12.53%)	154.93k (-15.93%)	159.45k (-13.48%)	163.30k (-11.39%)
	MoC-System	152.46k	105.24k (-30.97%)	84.43k (-44.62%)	64.15k (-57.92%)	73.36k (-51.88%)	93.17k (-38.89%)
	Lazarus	143.98k	115.34k (-19.89%)	109.68k (-23.82%)	100.23k (-30.39%)	102.83k (-28.58%)	113.28k (-21.32%)
	MoEvement	161.37k	122.51k (-24.08%)	106.70k (-33.88%)	90.32k (-44.03%)	94.48k (-41.45%)	117.82k (-26.99%)
	MeCeFO-v2	**203.73k**	**197.68k (-2.97%)**	**194.03k (-4.76%)**	**191.21k (-6.15%)**	**191.34k (-6.08%)**	**195.71k (-3.94%)**

Table 4: Zero-shot downstream evaluation of the Qwen3-MoE 16B-A2B model. Higher is better.

Scenario	BoolQ	ARC-E	ARC-C	PIQA	SciQ	HellaSwag	Avg.
Baseline	62.45	74.96	**41.92**	76.01	91.60	**61.31**	68.04
Sparse-Fail	62.14	75.94	40.81	75.19	91.30	61.04	67.74
Base-Fail	63.36	75.26	41.20	74.75	91.80	60.25	67.77
Frequent-Fail	62.05	74.15	40.89	**76.22**	91.80	60.43	67.59
Slow-Recover	**64.22**	74.57	41.64	75.57	92.10	60.83	68.16
Fast-Recover	62.87	**76.11**	41.68	75.78	**92.20**	60.76	**68.23**

also degrade consistently. These results show that the proposed strategy keeps training online while reducing the large recovery-time throughput penalty observed in prior systems. Under physical 8-GPU node failures, MeCeFO-v2 limits throughput degradation to 0.97–2.17% with at most a 0.17 PPL increase (Appendix D.1).

5.4 Downstream Task Evaluation

We further evaluate the trained Qwen3-MoE 16B-A2B checkpoint on six zero-shot downstream tasks: BoolQ (Clark et al., 2019), ARC-Easy/Challenge (Clark et al., 2018), PIQA (Bisk et al., 2019), SciQ (Welbl et al., 2017), and HellaSwag (Zellers et al., 2019). Table 4 shows that the downstream accuracy remains stable across all failure scenarios. The average score ranges from 67.59 to 68.23 (baseline: 68.04), i.e., every failure scenario stays within 0.45 points of the baseline, and Slow-Recover and Fast-Recover slightly exceed it. Individual tasks show similarly small fluctuations, with no consistent degradation under either more frequent failures or longer recovery. These results suggest that the approximate failover computation used by MeCeFO-v2 does not materially harm the semantic capabilities learned by the model. We also fine-tune the trained checkpoint on the Tulu-3-SFT-Mixture dataset (Lambert et al., 2024) and evaluate downstream performance after SFT in Appendix D.2. Additional ablations are in Appendix D.3.

6 Conclusions and Limitations

This paper presents MeCeFO-v2, a fault-tolerant optimization framework for sparse MoE training under hybrid PP-EP-DP parallelism. MeCeFO-v2 redistributes expert workloads across healthy EP peers and performs structured-approximation to maintain online training, through norm-guided head selection, router-aware sequence-level expert compression, and a budget-aware controller. Experiments show improved fault-time throughput over prior MoE fault-tolerance baselines and performance close to fault-free training.

A key limitation is that we focus on GPU failures, the most common failure mode in large-scale LLM training, and do not cover broader system failures such as CPU or storage failures. Extending MeCeFO-v2 to a more general system-level fault model is an important direction for future work.

AI USE STATEMENT

In this work, we used generative AI tools to aid or polish the writing. All AI-assisted text was reviewed and revised by the authors, who take full responsibility for the final content.

REFERENCES

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language, 2019. URL https://arxiv.org/abs/1911.11641.

Ruisi Cai, Yeonju Ro, Geon-Woo Kim, Peihao Wang, Babak E Bejnordi, Aditya Akella, and Zhangyang Wang. Read-ME: Refactorizing llms as router-decoupled mixture of experts with system co-design. *Advances in Neural Information Processing Systems*, 2024.

Weilin Cai, Le Qin, and Jiayi Huang. MoC-system: Efficient fault tolerance for sparse mixture-of-experts model training. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2025.

Runxi Cheng, Yuchen Guan, Yucheng Ding, Qingguo Hu, Yongxian Wei, Chun Yuan, Yelong Shen, Weizhu Chen, and Yeyun Gong. Mixture of neuron experts. *arXiv preprint arXiv:2510.05781*, 2025.

Minsik Choi, Hyegang Son, Changhoon Kim, and Young Geun Kim. Entropy meets importance: A unified head importance-entropy score for stable and efficient transformer pruning. *arXiv preprint arXiv:2510.13832*, 2025.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions, 2019. URL https://arxiv.org/abs/1905.10044.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1280–1297, 2024.

Abhinandan Das, Indranil Gupta, and Ashish Motivala. SWIM: Scalable weakly-consistent infection-style process group membership protocol. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*, pp. 303–312, 2002.

Jeffrey Dean and Luiz André Barroso. The tail at scale. *Communications of the ACM*, 2013.

Jianbo Dong, Bin Luo, Jun Zhang, Pengcheng Zhang, Fei Feng, Yikai Zhu, Ang Liu, Zian Chen, Yi Shi, Hairong Jiao, et al. Boosting large-scale parallel training efficiency with c4: A communication-driven approach. *arXiv preprint arXiv:2406.04594*, 2024.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International conference on machine learning*, pp. 5547–5569. PMLR, 2022.

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

Zizhuo Fu, Wenxuan Zeng, Runsheng Wang, and Meng Li. Attention sink forges native moe in attention layers: Sink-aware training to address head collapse. *arXiv preprint arXiv:2602.01203*, 2026.

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

Swapnil Gandhi and Christos Kozyrakis. Sparse checkpointing for fast and reliable MoE training, 2026. URL https://arxiv.org/abs/2412.15411.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Xiangming Gu, Tianyu Pang, Chao Du, Qian Liu, Fengzhuo Zhang, Cunxiao Du, Ye Wang, and Min Lin. When attention sink emerges in language models: An empirical view. In *International Conference on Learning Representations*, 2025.

Tianyu Guo, Druv Pai, Yu Bai, Jiantao Jiao, Michael I Jordan, and Song Mei. Active-dormant attention heads: Mechanistically demystifying extreme-token phenomena in LLMs. *arXiv preprint arXiv:2410.13835*, 2024a.

Yuxi Guo and Paul Sheridan. Greedy-gnorm: A gradient matrix norm-based alternative to attention entropy for head pruning. *arXiv preprint arXiv:2602.04491*, 2026.

Zhiyu Guo, Hidetaka Kamigaito, and Taro Watanabe. Attention score is not all you need for token importance indicator in kv cache reduction: Value also matters. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024b.

Rizhen Hu, Yutong He, Ran Yan, Mou Sun, Binghang Yuan, and Kun Yuan. MeCeFo: Enhancing llm training robustness via fault-tolerant optimization. *arXiv preprint arXiv:2510.16415*, 2025.

Insu Jang, Zhenning Yang, Zhen Zhang, Xin Jin, and Mosharaf Chowdhury. Oobleck: Resilient distributed training of large models using pipeline templates. In *Proceedings of the 29th Symposium on Operating Systems Principles*, 2023.

Ghadeer Jaradat, Mohammed Tolba, Ghada Alsuhli, Hani Saleh, Mahmoud Al-Qutayri, Thanos Stouraitis, and Baker Mohammad. Hybrid dynamic pruning: A pathway to efficient transformer inference. *arXiv preprint arXiv:2407.12893*, 2024.

Peng Jin, Bo Zhu, Li Yuan, and Shuicheng Yan. Moe++: Accelerating mixture-of-experts methods with zero-computation experts. In *International Conference on Learning Representations*, 2025.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

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

Jiaoda Li, Ryan Cotterell, and Mrinmaya Sachan. Differentiable subset pruning of transformer heads. *Transactions of the Association for Computational Linguistics*, 9:1442–1459, 2021.

Xinyu Lian, Sam Ade Jacobs, Lev Kurilenko, Masahiro Tanaka, Stas Bekman, Olatunji Ruwase, and Minjia Zhang. Universal checkpointing: A flexible and efficient distributed checkpointing system for large-scale dnn training with reconfigurable parallelism. In *2025 USENIX Annual Technical Conference (USENIX ATC 25)*, pp. 1519–1534, 2025.

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-V2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024a.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024b.

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

Avinash Maurya, Robert Underwood, M Mustafa Rafique, Franck Cappello, and Bogdan Nicolae. Datastates-llm: Lazy asynchronous checkpointing for large language models. In *Proceedings of the 33rd international symposium on high-performance parallel and distributed computing*, pp. 227–239, 2024.

Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? *Advances in neural information processing systems*, 2019.

Jayashree Mohan, Amar Phanishayee, and Vijay Chidambaram. CheckFreq: Frequent, Fine-Grained DNN checkpointing. In *USENIX Conference on File and Storage Technologies*, 2021.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 2024.

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

Pedro Sandoval-Segura, Xijun Wang, Ashwinee Panda, Micah Goldblum, Ronen Basri, Tom Goldstein, and David Jacobs. Identifying and evaluating inactive heads in pretrained LLMs. *arXiv preprint arXiv:2504.03889*, 2025.

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

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

Lalith Suresh, Dahlia Malkhi, Parikshit Gopalan, Ivan Porto Carreiro, and Zeeshan Lokhandwala. Stable and consistent membership at scale with Rapid. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pp. 387–401, 2018.

Zheyue Tan, Zhiyuan Li, Tao Yuan, Dong Zhou, Weilin Liu, Yueqing Zhuang, Yadong Li, Guowei Niu, Cheng Qin, Zhuyu Yao, et al. ReXMoE: Reusing experts with minimal overhead in mixture-of-experts. *arXiv preprint arXiv:2510.17483*, 2025.

John Thorpe, Pengzhan Zhao, Jonathan Eyolfson, Yifan Qiao, Zhihao Jia, Minjia Zhang, Ravi Netravali, and Guoqing Harry Xu. Bamboo: Making preemptible instances resilient for affordable training of large DNNs. In *USENIX Symposium on Networked Systems Design and Implementation*, 2023.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Guanhua Wang, Olatunji Ruwase, Bing Xie, and Yuxiong He. Fastpersist: Accelerating model checkpointing in deep learning. *arXiv preprint arXiv:2406.13768*, 2024.

Johannes Welbl, Nelson F Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 94–106, 2017.

Tiansheng Wen, Yifei Wang, Aosong Feng, Long Ma, Xinyang Liu, Yifan Wang, Lixuan Guo, Bo Chen, Stefanie Jegelka, and Chenyu You. Route experts by sequence, not by token. *arXiv preprint arXiv:2511.06494*, 2025.

Sunghyeon Woo, Baeseong Park, Byeongwook Kim, Minjung Jo, Se Jung Kwon, Dongsuk Jeon, and Dongsoo Lee. Dropbp: Accelerating fine-tuning of large language models by dropping backward propagation. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems*, 2024.

Baodong Wu, Lei Xia, Qingping Li, Kangyu Li, Xu Chen, Yongqiang Guo, Tieyao Xiang, Yuheng Chen, and Shigang Li. Transom: An efficient fault-tolerant system for training LLMs. *arXiv preprint arXiv:2310.10046*, 2023.

Tong Wu, Yutong He, Bin Wang, and Kun Yuan. Mixture-of-channels: Exploiting sparse ffns for efficient llms pre-training and inference. *arXiv preprint arXiv:2511.09323*, 2025.

Yongji Wu, Wenjie Qu, Xueshen Liu, Tianyang Tao, Yifan Qiao, Zhuang Wang, Wei Bai, Yuan Tian, Jiaheng Zhang, Z Morley Mao, et al. Lazarus: Resilient and elastic training of mixture-of-experts models. *arXiv preprint arXiv:2407.04656*, 2024.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, 2024.

Wenqian Xie, Zhiquan Lai, Shengwei Li, Weijie Liu, Wei Wang, Yanqi Hao, and Dongsheng Li. Capricorn: Efficient in-memory checkpointing for moe model training with dynamicity awareness. In *2025 IEEE International Conference on Cluster Computing*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019. URL https://arxiv.org/abs/1905.07830.

Zhengyan Zhang, Fanchao Qi, Zhiyuan Liu, Qun Liu, and Maosong Sun. Know what you don't need: single-shot meta-pruning for attention heads. *AI Open*, 2:36–42, 2021.

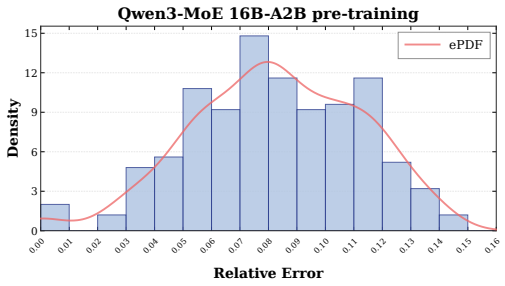


Figure 5: Single-batch relative error while pre-training Qwen3-MoE 16B-A2B.

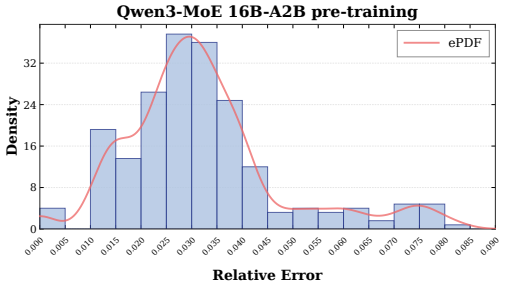


Figure 6: Full-batch relative error while pre-training Qwen3-MoE 16B-A2B.

A Convergence Analysis

This section establishes a nonconvex convergence proof for MeCeFO-v2 when using momentum SGD. The optimization problem is formulated as

$$\min_{\mathbf{w}} f(\mathbf{w}) := \frac{1}{D}\sum_{d=1}^{D} f_d(\mathbf{w}), \qquad f_d(\mathbf{w}) := \mathbb{E}_{\xi\sim\mathcal{D}_d}[F(\mathbf{w};\xi)],$$

where $\mathbf{w}$ collects all model parameters, $\mathcal{D}_d$ is the data distribution on DP replica d, and F is the per-sample loss. At iteration t, the global minibatch consists of n samples split evenly across the D replicas; sample $\xi_i^{(t)}$ is drawn independently from $\mathcal{D}_{d(i)}$, where $d(i)$ denotes the replica that processes it. Define the fault-free averaged stochastic gradient

$$\boldsymbol{g}_\star^{(t)} := \frac{1}{n}\sum_{i=1}^{n} \nabla F(\boldsymbol{w}^{(t)};\xi_i^{(t)}), \tag{1}$$

and denote the MeCeFO-v2 gradient under protection mode by $\widetilde{\boldsymbol{g}}^{(t)}$. The optimization update is momentum SGD:

$$\boldsymbol{m}^{(t)} = \beta_1 \boldsymbol{m}^{(t-1)} + (1-\beta_1)\widetilde{\boldsymbol{g}}^{(t)}, \tag{2}$$

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta \boldsymbol{m}^{(t)}, \tag{3}$$

with $\boldsymbol{m}^{(-1)} = \mathbf{0}$, $\beta_1 \in (0,1)$, and $\eta > 0$.

We begin with the following assumptions, where Assumption 2 and 3 are standard assumptions for convergence analysis, and Assumption 4 is observed empirically in our experiments.

Assumption 2 (Lower boundedness and smoothness)**.** *Function f is lower bounded and L-smooth:*

$$\inf_{\boldsymbol{w}\in\mathbb{R}^d} f(\boldsymbol{w}) > -\infty, \qquad \|\nabla f(\boldsymbol{w}) - \nabla f(\boldsymbol{w}')\|_2 \le L\|\boldsymbol{w}-\boldsymbol{w}'\|_2,\ \forall \boldsymbol{w}, \boldsymbol{w}'.$$

Assumption 3 (Stochastic gradients)**.** *There exists $\sigma > 0$ such that for each $i \in [n]$ and any $\boldsymbol{w}$,*

$$\mathbb{E}\big[\nabla F(\boldsymbol{w};\xi_i)\big] = \nabla f_{d(i)}(\boldsymbol{w}), \qquad \mathbb{E}\big[\|\nabla F(\boldsymbol{w};\xi_i) - \nabla f_{d(i)}(\boldsymbol{w})\|_2^2\big] \le \sigma^2.$$

Assumption 4 (Bounded approximation error)**.** *There exists $\underline{\delta} \in (0,1]$ such that for all iterations,*

$$\|\widetilde{\boldsymbol{g}}^{(t)} - \boldsymbol{g}_\star^{(t)}\|_2^2 \le (1-\underline{\delta})\|\boldsymbol{g}_\star^{(t)}\|_2^2, \tag{4}$$

$$\left\|\mathbb{E}\left[\widetilde{\boldsymbol{g}}^{(t)}\right] - \nabla f(\boldsymbol{w}^{(t)})\right\|_2^2 \le (1-\underline{\delta})\|\nabla f(\boldsymbol{w}^{(t)})\|_2^2. \tag{5}$$

Here and below, $\mathbb{E}[\widetilde{\boldsymbol{g}}^{(t)}]$ denotes the expectation over the minibatch sampling at iteration t, conditioned on the history up to $\boldsymbol{w}^{(t)}$.

Remark. Assumptions 2 and 3 are standard assumptions commonly used in convergence analysis. To validate Assumption 4, we empirically observe the relative errors through our experiments. Specifically, we observe the single-batch relative error $\|\widetilde{\boldsymbol{g}}^{(t)} - \boldsymbol{g}_\star^{(t)}\|_2^2 / \|\boldsymbol{g}_\star^{(t)}\|_2^2$ and full-batch relative error

$\|\mathbb{E}[\widetilde{\boldsymbol{g}}^{(t)}] - \nabla f(\boldsymbol{w}^{(t)})\|_2^2 / \|\nabla f(\boldsymbol{w}^{(t)})\|_2^2$ while pre-training Qwen3-MoE 16B-A2B. As illustrated in Fig. 5 and 6, these errors are consistently smaller than 0.15, justifying the application of Assumption 4.

Define

$$\Delta := f(\boldsymbol{w}^{(0)}) - \inf_{\boldsymbol{w}} f(\boldsymbol{w}), \tag{6}$$

$$\Delta_1 := \mathbb{E}\left[\|\boldsymbol{m}^{(0)} - \nabla f(\boldsymbol{w}^{(0)})\|_2^2\right]. \tag{7}$$

We present the formal version of the main theorem as follows:

Theorem 5 (Convergence of MeCeFO-v2)**.** *Under Assumptions 2, 3, and 4, if*

$$\beta_1 \in \left(1 - \frac{\underline{\delta}}{24 - 12\underline{\delta}},\, 1\right), \qquad \eta \le \min\left\{\frac{1}{2L}, \sqrt{\frac{\underline{\delta}(1-\beta_1)^2}{8L^2}}\right\}, \tag{8}$$

then

$$\frac{1}{T+1}\sum_{t=0}^{T}\mathbb{E}\left[\|\nabla f(\boldsymbol{w}^{(t)})\|_2^2\right] \le \frac{8\Delta}{\underline{\delta}\eta(T+1)} + \frac{8\Delta_1}{\underline{\delta}(1-\beta_1)(T+1)} + \frac{24(1-\beta_1)\sigma^2}{\underline{\delta}n}. \tag{9}$$

To prove the main theorem, we need the following lemmas:

Lemma 6 (Descent lemma)**.** *Under Assumption 2,*

$$f(\boldsymbol{w}^{(t+1)}) \le f(\boldsymbol{w}^{(t)}) - \left(\frac{1}{2\eta} - \frac{L}{2}\right)\|\boldsymbol{w}^{(t+1)} - \boldsymbol{w}^{(t)}\|_2^2 \tag{10}$$

$$+ \frac{\eta}{2}\|\nabla f(\boldsymbol{w}^{(t)}) - \boldsymbol{m}^{(t)}\|_2^2 - \frac{\eta}{2}\|\nabla f(\boldsymbol{w}^{(t)})\|_2^2. \tag{11}$$

Proof. By L-smoothness,

$$f(\boldsymbol{w}^{(t+1)}) \le f(\boldsymbol{w}^{(t)}) + \langle \nabla f(\boldsymbol{w}^{(t)}), \boldsymbol{w}^{(t+1)} - \boldsymbol{w}^{(t)}\rangle + \frac{L}{2}\|\boldsymbol{w}^{(t+1)} - \boldsymbol{w}^{(t)}\|_2^2.$$

Rewrite the inner product by adding and subtracting $\boldsymbol{m}^{(t)}/2$ and use $\boldsymbol{w}^{(t+1)} - \boldsymbol{w}^{(t)} = -\eta \boldsymbol{m}^{(t)}$. □

Lemma 7 (Momentum-gradient gap)**.** *Under Assumptions 2, 3, and 4,*

$$\begin{aligned}
&\frac{1}{T+1}\sum_{t=0}^{T}\mathbb{E}\left[\|\boldsymbol{m}^{(t)} - \nabla f(\boldsymbol{w}^{(t)})\|_2^2\right] \\
\le\ & \frac{2\Delta_1}{(1-\beta_1)(T+1)} + \frac{4L^2}{\underline{\delta}(1-\beta_1)^2}\cdot\frac{1}{T+1}\sum_{t=1}^{T}\mathbb{E}\left[\|\boldsymbol{w}^{(t)} - \boldsymbol{w}^{(t-1)}\|_2^2\right] \\
&+ \left(1 - \frac{\underline{\delta}}{2}\right)(7 - 6\beta_1)\cdot\frac{1}{T+1}\sum_{t=1}^{T}\mathbb{E}\left[\|\nabla f(\boldsymbol{w}^{(t)})\|_2^2\right] + \frac{6(1-\beta_1)\sigma^2}{n}.
\end{aligned} \tag{12}$$

Proof. For $t \ge 1$, from equation 2,

$$\begin{aligned}
&\mathbb{E}\left[\|\boldsymbol{m}^{(t)} - \nabla f(\boldsymbol{w}^{(t)})\|_2^2\right] \\
=&\mathbb{E}\left[\left\|\beta_1(\boldsymbol{m}^{(t-1)} - \nabla f(\boldsymbol{w}^{(t)})) + (1-\beta_1)(\widetilde{\boldsymbol{g}}^{(t)} - \nabla f(\boldsymbol{w}^{(t)}))\right\|_2^2\right] \\
=&\mathbb{E}\left[\left\|\beta_1(\boldsymbol{m}^{(t-1)} - \nabla f(\boldsymbol{w}^{(t)})) + (1-\beta_1)(\mathbb{E}[\widetilde{\boldsymbol{g}}^{(t)}] - \nabla f(\boldsymbol{w}^{(t)}))\right\|_2^2\right] \\
&+ (1-\beta_1)^2\mathbb{E}\left[\|\widetilde{\boldsymbol{g}}^{(t)} - \mathbb{E}[\widetilde{\boldsymbol{g}}^{(t)}]\|_2^2\right].
\end{aligned} \tag{13}$$

The first term is bounded by Jensen and equation 5:

$$\begin{aligned}
&\mathbb{E}\left[\left\|\beta_1(\boldsymbol{m}^{(t-1)} - \nabla f(\boldsymbol{w}^{(t)})) + (1-\beta_1)(\mathbb{E}[\widetilde{\boldsymbol{g}}^{(t)}] - \nabla f(\boldsymbol{w}^{(t)}))\right\|_2^2\right] \\
\le&\beta_1\mathbb{E}\left[\|\boldsymbol{m}^{(t-1)} - \nabla f(\boldsymbol{w}^{(t)})\|_2^2\right] + (1-\underline{\delta})(1-\beta_1)\mathbb{E}\left[\|\nabla f(\boldsymbol{w}^{(t)})\|_2^2\right].
\end{aligned} \tag{14}$$

By Young's inequality,

$$\mathbb{E}\left[\|\boldsymbol{m}^{(t-1)} - \nabla f(\boldsymbol{w}^{(t)})\|_2^2\right] \leq \left(1 + \frac{\underline{\delta}(1-\beta_1)}{2}\right)\mathbb{E}\left[\|\boldsymbol{m}^{(t-1)} - \nabla f(\boldsymbol{w}^{(t-1)})\|_2^2\right] + \left(1 + \frac{2}{\underline{\delta}(1-\beta_1)}\right)\mathbb{E}\left[\|\nabla f(\boldsymbol{w}^{(t)}) - \nabla f(\boldsymbol{w}^{(t-1)})\|_2^2\right]. \quad (15)$$

For the variance term in equation 13,

$$\begin{aligned}
&\mathbb{E}\left[\|\widetilde{\boldsymbol{g}}^{(t)} - \mathbb{E}[\widetilde{\boldsymbol{g}}^{(t)}]\|_2^2\right] \\
\leq& 3\mathbb{E}\left[\|\widetilde{\boldsymbol{g}}^{(t)} - \boldsymbol{g}_\star^{(t)}\|_2^2\right] + 3\mathbb{E}\left[\|\boldsymbol{g}_\star^{(t)} - \nabla f(\boldsymbol{w}^{(t)})\|_2^2\right] + 3\mathbb{E}\left[\|\nabla f(\boldsymbol{w}^{(t)}) - \mathbb{E}[\widetilde{\boldsymbol{g}}^{(t)}]\|_2^2\right] \\
\leq& 3(1-\underline{\delta})\mathbb{E}\left[\|\boldsymbol{g}_\star^{(t)}\|_2^2\right] + \frac{3\sigma^2}{n} + 3(1-\underline{\delta})\mathbb{E}\left[\|\nabla f(\boldsymbol{w}^{(t)})\|_2^2\right] \\
\leq& 6(1-\underline{\delta})\mathbb{E}\left[\|\nabla f(\boldsymbol{w}^{(t)})\|_2^2\right] + \frac{(6-3\underline{\delta})\sigma^2}{n}, \quad (16)
\end{aligned}$$

where we used equation 4, equation 5, and $\mathbb{E}[\|\boldsymbol{g}_\star^{(t)} - \nabla f(\boldsymbol{w}^{(t)})\|_2^2] \leq \sigma^2/n$.

Combining equation 13–equation 16 and smoothness,

$$\begin{aligned}
\mathbb{E}\left[\|\boldsymbol{m}^{(t)} - \nabla f(\boldsymbol{w}^{(t)})\|_2^2\right] \leq& \left(1 - (1-\beta_1)\left(1 - \frac{\underline{\delta}}{2}\right)\right)\mathbb{E}\left[\|\boldsymbol{m}^{(t-1)} - \nabla f(\boldsymbol{w}^{(t-1)})\|_2^2\right] \\
&+ \frac{2L^2}{\underline{\delta}(1-\beta_1)}\mathbb{E}\left[\|\boldsymbol{w}^{(t)} - \boldsymbol{w}^{(t-1)}\|_2^2\right] \\
&+ (1-\underline{\delta})(1-\beta_1)(7-6\beta_1)\mathbb{E}\left[\|\nabla f(\boldsymbol{w}^{(t)})\|_2^2\right] \\
&+ \frac{(6-3\underline{\delta})(1-\beta_1)^2\sigma^2}{n}. \quad (17)
\end{aligned}$$

Summing equation 17 from $t = 1$ to T yields equation 12. □

Now we can start our proof of Theorem 5.

Proof of Theorem 5. Summing equation 11 for $t = 0, \ldots, T$ and taking expectation:

$$\begin{aligned}
\inf_{\boldsymbol{w}} f(\boldsymbol{w}) - f(\boldsymbol{w}^{(0)}) \leq& \frac{\eta}{2}\sum_{t=0}^{T}\mathbb{E}\left[\|\nabla f(\boldsymbol{w}^{(t)}) - \boldsymbol{m}^{(t)}\|_2^2\right] \\
&- \left(\frac{1}{2\eta} - \frac{L}{2}\right)\sum_{t=0}^{T}\mathbb{E}\left[\|\boldsymbol{w}^{(t+1)} - \boldsymbol{w}^{(t)}\|_2^2\right] \\
&- \frac{\eta}{2}\sum_{t=0}^{T}\mathbb{E}\left[\|\nabla f(\boldsymbol{w}^{(t)})\|_2^2\right]. \quad (18)
\end{aligned}$$

Apply Lemma 7 to equation 18. Since $\beta_1 > 1 - \underline{\delta}/(24 - 12\underline{\delta})$ implies $(1-\underline{\delta}/2)(7-6\beta_1) \leq 1 - \underline{\delta}/4$, we get

$$\begin{aligned}
\frac{1}{T+1}\sum_{t=0}^{T}\mathbb{E}\left[\|\nabla f(\boldsymbol{w}^{(t)})\|_2^2\right] \leq& -\frac{8}{\underline{\delta}\eta}\left(\frac{1}{2\eta} - \frac{L}{2} - \frac{2\eta L^2}{\underline{\delta}(1-\beta_1)^2}\right)\frac{1}{T+1}\sum_{t=0}^{T}\mathbb{E}\left[\|\boldsymbol{w}^{(t+1)} - \boldsymbol{w}^{(t)}\|_2^2\right] \\
&+ \frac{8\Delta}{\underline{\delta}\eta(T+1)} + \frac{8\Delta_1}{\underline{\delta}(1-\beta_1)(T+1)} + \frac{24(1-\beta_1)\sigma^2}{\underline{\delta}n}. \quad (19)
\end{aligned}$$

Condition equation 8 makes the coefficient of $\sum\|\boldsymbol{w}^{(t+1)} - \boldsymbol{w}^{(t)}\|_2^2$ nonnegative, so that term is dropped, yielding equation 9. □

The convergence rate can be derived immediately as the following corollary.

Table 5: Architecture details of the Qwen3-MoE models used in our experiments. "Activated experts" denotes the number of routed experts selected per token.

Model	Experts	Activated experts	Layers	Hidden size	Attention heads
Qwen3-MoE 1B-A0.4B	16	4	16	1024	16
Qwen3-MoE 8B-A1.3B	96	8	24	1536	12
Qwen3-MoE 16B-A2B	128	8	24	2048	16

Corollary 8 (Rate of MeCeFO-v2)**.** *Under Theorem 5, choose*

$$\eta = \left(2L + \sqrt{\frac{8L^2}{\underline{\delta}(1-\beta_1)^2}}\right)^{-1}, \qquad \beta_1 = 1 - \left(\frac{24}{\underline{\delta}} + \sqrt{\frac{\underline{\delta}^{1/2}(T+1)\sigma^2}{n(L\Delta + \underline{\delta}^{1/2}\Delta_1)}}\right)^{-1}. \tag{20}$$

Then

$$\frac{1}{T+1}\sum_{t=0}^{T}\mathbb{E}\left[\|\nabla f(\boldsymbol{w}^{(t)})\|_2^2\right] = \mathcal{O}\left(\sqrt{\frac{(L\Delta + \underline{\delta}^{1/2}\Delta_1)\sigma^2}{\underline{\delta}^{5/2}n(T+1)}} + \frac{L\Delta + \underline{\delta}^{1/2}\Delta_1}{\underline{\delta}^{5/2}(T+1)}\right). \tag{21}$$

Hence MeCeFO-v2 preserves the standard distributed nonconvex order $\mathcal{O}(1/\sqrt{nT})$.

Proof. Substitute the displayed choices of (η, β_1) into Theorem 5 and simplify the two dominant terms with respect to T, n, and $\underline{\delta}$. □

B Method Details

This appendix expands the resource model summarized in Sec. 4.4 and the routing-score analysis in Sec. 4.3. These details are not needed to follow the main design, but they make the feasibility test fully reproducible. System-level implementation details, including protection-mode execution, are given in Appendix C.

B.1 Model architecture details

Our experiments instantiate MeCeFO-v2 on Qwen3-MoE-style backbones (Yang et al., 2025). Although the main text describes the attention module using conventional multi-head attention (MHA) notation for simplicity, the Qwen3 models used in our experiments employ grouped-query attention (GQA). Specifically, let H_ℓ and G_ℓ denote the numbers of query heads and KV heads at layer ℓ, respectively, with $G_\ell < H_\ell$. Each query head $h \in [H_\ell]$ is associated with a KV group $\phi_\ell(h) \in [G_\ell]$, and multiple query heads share the same key-value representations:

$$\mathbf{O}_{\ell,h} = \mathrm{Softmax}\Big(\mathbf{Q}_{\ell,h}\mathbf{K}_{\ell,\phi_\ell(h)}^\top/\sqrt{d}\Big)\,\mathbf{V}_{\ell,\phi_\ell(h)}.$$

This modification does not conflict with our norm-based head-selection mechanism. In the GQA setting, since multiple query heads share the same key-value projection, MeCeFO-v2 performs selection at the KV-head group level rather than independently for each query head. We compute the importance score of each KV group by aggregating the value-output norms of all query heads associated with it, and retain the groups with the largest scores. The backward pass is preserved for all query heads in retained KV groups and skipped for those in discarded groups, ensuring consistent treatment of shared key-value projections.

We evaluate across different model scales, as summarized in Table 5. The three models share the same top-K MoE routing pattern (Shazeer et al., 2017) within each sparse layer but differ in total expert capacity, activated expert count, depth, hidden width, and key-value head count. These configurations cover a range of expert-parallel pressure points: the smaller Qwen3-MoE 1B-A0.4B model provides a lightweight validation setting, while the Qwen3-MoE 8B-A1.3B and Qwen3-MoE 16B-A2B variants increase both the number of experts and the hidden dimension, stressing the redistribution and compression policies used during protection-mode training.

B.2 MODEL TRAINING DETAILS

All models are trained on the FineWeb-Edu dataset (Penedo et al., 2024) for 50B tokens using a Megatron-based (Shoeybi et al., 2019) training codebase. We adopt a standard Transformer training configuration with sequence length 1024 and global batch size 2048. Optimization uses AdamW (Loshchilov & Hutter, 2019) with weight decay 0.1, gradient clipping at 1.0, and coefficients $(\beta_1, \beta_2) = (0.9, 0.95)$. The coefficient of the load balancing loss is set to 1×10^{-3}. The peak learning rate is 4×10^{-4} for the Qwen3-MoE 1B-A0.4B model and 3×10^{-4} for the Qwen3-MoE 8B-A1.3B and 16B-A2B models. For all model scales, the learning rate is linearly warmed up during the first 1500 iterations and then decayed with a cosine schedule to 10% of the peak learning rate.

Training uses hybrid parallelism with data parallelism, expert parallelism, and pipeline parallelism degrees set to $\mathrm{DP} = 8$, $\mathrm{EP} = 4$, and $\mathrm{PP} = 4$, respectively. For MoE compression, we set $T_{\mathrm{SVD}} = 100$, i.e., the low-rank projection matrices are updated by SVD every 100 training iterations. For budget-aware automatic parameter selection, we profile twice during training, at iterations 1500 and 12500, which is sufficient to obtain strong policies in our experiments while adding negligible overhead relative to the full training run.

B.3 RESOURCE MODEL FOR POLICY SELECTION

In this section, we provide a concrete resource model used by the controller to select $\theta = (\kappa, \chi)$ under a simplified theoretical memory and compute constraints. For each affected stage, the controller searches over $\theta = (\kappa, \chi)$, where κ is the retained attention-head budget and χ is the average number of compressed expert assignments per token. We normalize these as

$$\alpha(\kappa) := \frac{\kappa}{H}, \qquad q(\chi) := \frac{\chi}{K},$$

where H is the number of attention heads per layer and K is the number of activated experts per token. All layers of a stage share the same H and K in our models, so we drop the layer index. We write α and q for $\alpha(\kappa)$ and $q(\chi)$ when no ambiguity arises. All θ-independent memory and compute are grouped into M_p^{fix} and C_p^{fix}, including parameters, optimizer states, communication buffers, redistributed expert states, and activations or operations that remain necessary under any protection policy. The controllable terms below are scaled by $\lambda_E \geq 1$, the workload amplification factor of a protected rank induced by the redistribution in Sec. 4.1.

Memory model. We decompose the controllable memory into attention and MoE components. For attention layers, let A_p^{att} denote the number of attention activation elements cached by stage p under full retention. It can be obtained from runtime statistics or approximated as

$$A_p^{\mathrm{att}} \approx \sum_{\ell \in \mathcal{L}_p^{\mathrm{att}}} B(2s^2 H + 4sh),$$

where $\mathcal{L}_p^{\mathrm{att}}$ is the set of attention layers on stage p, B is the microbatch size, s is the sequence length, and h is the hidden width. With μ denoting the number of bytes per activation element, the retained attention cache scales linearly with α:

$$\widetilde{M}_p^{\mathrm{att}}(\kappa) = \mu \alpha A_p^{\mathrm{att}}.$$

For MoE layers, let M_p^{moe} denote the full MoE activation-cache size in bytes and let $\eta_p \in [0, 1]$ be the ratio between compressed (low-rank) and full activation cost, obtained from profiling. Since a fraction q of assignments is compressed, we model

$$\widetilde{M}_p^{\mathrm{moe}}(\chi) = (1 - q + q\eta_p) M_p^{\mathrm{moe}}.$$

The total protected-stage memory is

$$\mathrm{Mem}_p(\theta) = M_p^{\mathrm{fix}} + \lambda_E \big(\widetilde{M}_p^{\mathrm{att}}(\kappa) + \widetilde{M}_p^{\mathrm{moe}}(\chi) \big). \tag{22}$$

Compute model. MeCeFO-v2 keeps the forward pass exact and reduces computation only in selected backward terms. Let C_p^{att} and C_p^{moe} denote the forward costs of attention and MoE components, respectively. Then the effective compute is modeled as

$$\widetilde{C}_p^{\mathrm{att}}(\kappa) = (1 + 2\alpha) C_p^{\mathrm{att}}, \qquad \widetilde{C}_p^{\mathrm{moe}}(\chi) = (3 - q + q\beta_p) C_p^{\mathrm{moe}},$$

where $\beta_p \in [0, 1]$ captures the relative cost of compressed expert backward computation.

The total protected-stage compute is

$$\mathrm{Comp}_p(\theta) = C_p^{\mathrm{fix}} + \lambda_E\big(\widetilde{C}_p^{\mathrm{att}}(\kappa) + \widetilde{C}_p^{\mathrm{moe}}(\chi)\big). \tag{23}$$

The controller restricts the search to policies satisfying

$$\mathrm{Mem}_p(\theta) \le M_{\max}, \qquad \mathrm{Comp}_p(\theta) \le \mathrm{Comp}_p^{\mathrm{base}}.$$

B.4 Routing scores and expert-gradient compression error

This subsection derives the bound in Sec. 4.3. Fix a MoE layer and an expert j. For a token $\mathbf{x}$, the MoE output is $\mathbf{o}(\mathbf{x}) = \sum_{i\in\mathcal{A}(\mathbf{x})} \pi_i(\mathbf{x})E_i(\mathbf{x})$, where $\pi_i(\mathbf{x}) = \mathbf{z}(\mathbf{x})_i/S(\mathbf{x})$ and $S(\mathbf{x}) := \sum_{i\in\mathcal{A}(\mathbf{x})} \mathbf{z}(\mathbf{x})_i \in (0, 1]$. Let $\mathbf{g}(\mathbf{x}) := \partial\mathcal{L}/\partial\mathbf{o}(\mathbf{x})$. Consider a linear sublayer $\mathbf{y} = \mathbf{a}\mathbf{W}^\top$ inside E_j, with input row $\mathbf{a}_j(\mathbf{x})$ and output row $\mathbf{y}_j(\mathbf{x})$ for token $\mathbf{x}$; stacking these rows over the tokens routed to j gives $\mathbf{X}$ and $\mathbf{Y}$ in Sec. 4.3.

Lemma 9. *Let* $\mathbf{J}_j(\mathbf{x}) := \partial E_j(\mathbf{x})/\partial\mathbf{y}_j(\mathbf{x})$ *and* $\bar{\boldsymbol{\delta}}_j(\mathbf{x}) := \mathbf{g}(\mathbf{x})\mathbf{J}_j(\mathbf{x})$. *Then* $\partial\mathcal{L}/\partial\mathbf{y}_j(\mathbf{x}) = \pi_j(\mathbf{x})\bar{\boldsymbol{\delta}}_j(\mathbf{x})$, *where* $\bar{\boldsymbol{\delta}}_j(\mathbf{x})$ *does not contain the factor* $\pi_j(\mathbf{x})$.

Proof. The gate weight $\pi_j(\mathbf{x})$ depends only on the router parameters and $\mathbf{x}$, and $E_j(\mathbf{x})$ enters $\mathbf{o}(\mathbf{x})$ only through $\pi_j(\mathbf{x})E_j(\mathbf{x})$; hence $\partial\mathcal{L}/\partial E_j(\mathbf{x}) = \pi_j(\mathbf{x})\mathbf{g}(\mathbf{x})$. Backpropagation from $E_j(\mathbf{x})$ to $\mathbf{y}_j(\mathbf{x})$ is right-multiplication by $\mathbf{J}_j(\mathbf{x})$, which is evaluated at the forward pass of E_j and does not involve $\pi_j(\mathbf{x})$. By linearity, the scalar $\pi_j(\mathbf{x})$ factors out. □

Stacking rows gives $\boldsymbol{\Delta} = \boldsymbol{\Pi}_j\bar{\boldsymbol{\Delta}}$, where $\boldsymbol{\Pi}_j = \mathrm{diag}(\pi_j(\mathbf{x}))$ and the rows of $\bar{\boldsymbol{\Delta}}$ are $\bar{\boldsymbol{\delta}}_j(\mathbf{x})$; this is the gate-free factorization used in Sec. 4.3.

Per-assignment identity. Assignment $(\mathbf{x}, j)$ contributes the rank-one term $\pi_j(\mathbf{x})\bar{\boldsymbol{\delta}}_j(\mathbf{x})^\top\mathbf{a}_j(\mathbf{x})$ to $\mathbf{G}_W = \boldsymbol{\Delta}^\top\mathbf{X}$, and compression replaces $\mathbf{a}_j(\mathbf{x})$ by $\mathbf{a}_j(\mathbf{x})\mathbf{V}_{j,r}\mathbf{V}_{j,r}^\top$. Since $\|\mathbf{u}^\top\mathbf{v}\|_F = \|\mathbf{u}\|_2\|\mathbf{v}\|_2$ for row vectors, the resulting error is

$$\begin{aligned}\big\|\Delta\mathbf{G}_W(\mathbf{x}, j)\big\|_F &= \pi_j(\mathbf{x})\,\|\bar{\boldsymbol{\delta}}_j(\mathbf{x})\|_2\,\big\|\mathbf{a}_j(\mathbf{x})(\mathbf{I} - \mathbf{V}_{j,r}\mathbf{V}_{j,r}^\top)\big\|_2 \\ &= \mathbf{z}(\mathbf{x})_j \cdot \frac{\|\bar{\boldsymbol{\delta}}_j(\mathbf{x})\|_2}{S(\mathbf{x})} \cdot \big\|\mathbf{a}_j(\mathbf{x})(\mathbf{I} - \mathbf{V}_{j,r}\mathbf{V}_{j,r}^\top)\big\|_2.\end{aligned} \tag{24}$$

Matrix bound. Let $\boldsymbol{\Pi}_\mathcal{C}$, $\bar{\boldsymbol{\Delta}}_\mathcal{C}$, and $\mathbf{X}_\mathcal{C}$ denote the restrictions of $\boldsymbol{\Pi}_j$, $\bar{\boldsymbol{\Delta}}$, and $\mathbf{X}$ to the rows of the compressed assignments of expert j. Since the remaining assignments are computed exactly,

$$\mathbf{G}_W - \widetilde{\mathbf{G}}_W = \bar{\boldsymbol{\Delta}}_\mathcal{C}^\top\boldsymbol{\Pi}_\mathcal{C}\mathbf{X}_\mathcal{C}(\mathbf{I} - \mathbf{V}_{j,r}\mathbf{V}_{j,r}^\top).$$

Using $\|\mathbf{AB}\|_F \le \|\mathbf{A}\|_F\|\mathbf{B}\|_2$ and $\|\boldsymbol{\Pi}_\mathcal{C}\|_2 = \max\{\pi_j(\mathbf{x}) : (\mathbf{x}, j) \text{ is compressed}\} = \pi_j$, we obtain

$$\|\mathbf{G}_W - \widetilde{\mathbf{G}}_W\|_F \le \|\bar{\boldsymbol{\Delta}}_\mathcal{C}\|_F\,\|\boldsymbol{\Pi}_\mathcal{C}\|_2\,\|\mathbf{X}_\mathcal{C}(\mathbf{I} - \mathbf{V}_{j,r}\mathbf{V}_{j,r}^\top)\|_2 = \pi_j\,\|\bar{\boldsymbol{\Delta}}_\mathcal{C}\|_F\,\|\mathbf{X}_\mathcal{C}(\mathbf{I} - \mathbf{V}_{j,r}\mathbf{V}_{j,r}^\top)\|_2,$$

which is the bound stated in Sec. 4.3.

Discussion and empirical verification. The identity in equation 24 shows that the routing score used by the sequence-level selection rule in Sec. 4.3 is an explicit multiplicative factor of the compression error. Within a token, all selected experts share $S(\mathbf{x})$, so ranking by $\mathbf{z}(\mathbf{x})_j$ coincides with ranking by $\pi_j(\mathbf{x})$. The identity does not by itself guarantee that lower-score assignments always incur smaller error, because the remaining factors $\|\bar{\boldsymbol{\delta}}_j(\mathbf{x})\|_2/S(\mathbf{x})$ and the out-of-subspace residual of $\mathbf{a}_j(\mathbf{x})$ vary across tokens and experts. We therefore verify the trend empirically in Fig. 3 (right), where the error of each assignment is normalized by the full-layer gradient norm and assignments are grouped by their within-token router-score rank. Under sequence-level compression, the mean relative error decreases from about 1.7% for rank-1 assignments to about 0.45% for rank-4 assignments, and the same decreasing trend holds in all six layer blocks (layers 0–3 through 20–23) of Qwen3-MoE 16B-A2B.

C System Implementation Details

This appendix describes how MeCeFO-v2 detects failures, recovers the failed state, executes protection mode, and rejoins recovered ranks, and how the failover overhead is accounted for in our throughput measurements.

C.1 Failure detection and coordination

Following Lazarus (Wu et al., 2024), our prototype adopts a centralized controller–agent architecture. The controller manages the cluster, and an agent on each node launches the worker processes that run the MeCeFO-v2 runtime and communicates with the controller over TCP. The agents send a heartbeat for every GPU rank to the controller once per second, which allows the controller to maintain a global view of rank availability and to detect lost ranks. A rank is marked as failed after three consecutive missed heartbeats, and the controller then broadcasts the set of failed ranks F to the surviving workers. Based on the remaining ranks, the controller recomputes the expert placement, i.e., the balanced maps $\varphi_{\ell,F}$ in Sec. 4.1, and forwards the new configuration to the workers through the agents. The workers then rebuild their communication groups and enter protection mode.

The centralized controller is a control-plane choice rather than an algorithmic requirement: the MeCeFO-v2 runtime only consumes an agreed failed-rank set. If fault tolerance of the controller itself is required, the same interface can be provided by decentralized membership protocols such as SWIM (Das et al., 2002) or Rapid (Suresh et al., 2018), without changing the workload redistribution, compression, or optimization mechanisms of MeCeFO-v2.

C.2 State recovery

For a failed rank (d, p, e), the rank at the same position (p, e) in another DP replica holds the same model shard together with its optimizer states. After the balanced map assigns the failed experts to surviving EP ranks, each protected rank fetches from that replica only the parameters and Adam states (first moment, second moment, and step count) of the experts assigned to it. This transfer is performed once per failure rather than every iteration, and its cost is included in the end-to-end throughput. During protection mode, the DP averaging rule is unchanged, so the parameters and Adam states held by the replica are updated every iteration with the same reduced gradients and always remain up to date.

C.3 Protection-mode execution

The runtime handles a failure as follows. It first fetches the failed shard's states from an alive DP replica, builds balanced maps $\varphi_{\ell,F}$ for all failed-stage MoE layers, and marks the surviving EP ranks of the affected EP groups (e.g., $\mathcal{S}^{-f}_{d_f,p_f}$ for a single failed rank) as protected. Protected forward execution keeps layer outputs exact while changing the activation caches: retained attention heads keep their head-local caches, and selected MoE assignments cache low-rank projected activations. Protected backward execution backpropagates through retained heads, computes exact Dgrad for all routed expert assignments, and uses approximate Wgrad only for assignments in $\mathcal{C}_{\ell,q}(\chi)$. The policy is refreshed every T_{prof} steps by re-profiling gradient distortion and re-solving. Algorithms 2 and 3 give the protected forward and backward passes.

C.4 Pipeline execution and data ordering

Pipeline execution requires no replay or data reloading. Training data are stored in a distributed storage service independent of the compute GPUs, and the dataloader maintains the same global sample order regardless of failures. Consequently, even if a failure occurs at pipeline stage 1, no training samples are lost or reordered: after reconfiguration, the protected ranks continue processing the same global training stream in its original order.

Algorithm 2 MeCeFO-v2 protected forward pass

def `MeCeFOv2_Forward`$(\ell, \theta^\star = (\kappa, \chi))$

1: Execute the standard forward computation of layer ℓ to preserve the exact output.
2: **If the current rank is protected:**
3: **Attention layer:**
4: Compute $s_{\ell,h} = \|\mathbf{O}_{\ell,h}\|_F^2$ and $\mathcal{H}_\ell^\star(\kappa)$, cache head-activations only for $h \in \mathcal{H}_\ell^\star(\kappa)$.
5: **MoE layer:**
6: Route tokens with the top-K_ℓ router and keep expert forward computation exact.
7: **Per sequence** q**:** Construct $\mathcal{C}_{\ell,q}(\chi)$ and $\mathcal{F}_{\ell,q}(\chi)$.
8: Cache full activations $\mathbf{X}$ for $(\mathbf{x}, j) \in \mathcal{F}_{\ell,q}(\chi)$ and projected activations $\widehat{\mathbf{X}} = \mathbf{X}\mathbf{V}_{\ell,j,r_\ell}^{(t)}$ for $(\mathbf{x}, j) \in \mathcal{C}_{\ell,q}(\chi)$.

Algorithm 3 MeCeFO-v2 protected backward pass

def `MeCeFOv2_Backward`$(\ell, \theta^\star = (\kappa, \chi))$

1: **If the current rank is protected:**
2: **Attention layer:**
3: Backpropagate through heads in $\mathcal{H}_\ell^\star(\kappa)$ and stop gradients for discarded heads.
4: **MoE layer:**
5: Compute Dgrad exactly for all routed assignments.
6: Compute exact Wgrad $\mathbf{G}_W = \mathbf{\Delta}^\top \mathbf{X}$ for $(\mathbf{x}, j) \in \mathcal{F}_{\ell,q}(\chi)$.
7: Compute approximate Wgrad $\widetilde{\mathbf{G}}_W = \mathbf{\Delta}^\top \widehat{\mathbf{X}} \mathbf{V}_{\ell,j,r_\ell}^{(t)\top}$ for assignments in $\mathcal{C}_{\ell,q}(\chi)$.
8: Return the protected gradient $\widetilde{\mathbf{G}}_{d,\ell}^{(t)}$.

C.5 Rejoining and consistency

When the failed rank, or a replacement rank, becomes available again, it rejoins at the next iteration boundary by fetching the latest parameters and optimizer states together from the same DP replica. The original expert placement is then restored and protection mode ends. Because the parameters and optimizer states are taken from the same synchronization point, they are mutually consistent even after many protected updates, and no additional synchronization is required.

C.6 Throughput accounting and failover overhead

For every method, throughput is computed as the total number of completed training tokens divided by the total wall-clock time over the entire failure trace. It therefore includes failure detection, pipeline synchronization, communication-group reconfiguration, parameter and optimizer-state transfer, expert redistribution, controller preparation, kernel and collective warm-up, and all subsequent protected training. All compared methods use the same failure detector and the same accounting.

To quantify the transition cost, Table 6 reports the iteration time around a failure. The first measurement spans from the failure occurrence to the completion of the first iteration and thus captures the complete one-time transition cost; the following iterations show how quickly execution returns to steady state. The first iteration takes 12.76 s, and by the third iteration the iteration time decreases to 3.55 s, within 2.9% of the steady-state 3.45 s. The transition overhead is therefore concentrated in the first iteration, and it is fully included in the reported end-to-end throughput.

C.7 Interaction with checkpointing

MeCeFO-v2 complements rather than replaces checkpointing. For the failures considered in this work, the missing parameters and optimizer states are recovered directly from a healthy DP replica, provided that at least one replica of the affected state remains available. Checkpoint recovery is required only when all replicas of the affected state are lost before recovery, e.g., under correlated failures, which is much less likely than the failures targeted by MeCeFO-v2 under failure-domain-aware placement.

Table 6: Iteration time around a failure. Iteration 1 spans from the failure occurrence to the completion of the first iteration after the failure.

	No fault	Iter. 1	Iter. 2	Iter. 3	Iter. 4	Iter. 5	Steady state
Time (s)	3.43	12.76	3.82	3.55	3.46	3.45	3.45

Table 7: Throughput (token/s) and final validation PPL under single-GPU and node-level failures with failure-domain-aware placement ($D = 2$, $P = 4$, $E = 16$). The average number of unavailable GPUs matches the BASE-FAIL scenario.

Model	No fault		Single-GPU failures		Node-level failures	
	Throughput	PPL	Throughput	PPL	Throughput	PPL
Qwen3-MoE 1B-A0.4B	526.06k	12.70	514.24k	12.82	518.62k	12.87
Qwen3-MoE 8B-A1.3B	248.71k	9.21	244.53k	9.27	246.29k	9.30
Qwen3-MoE 16B-A2B	185.38k	8.47	180.06k	8.52	181.35k	8.53

Because MeCeFO-v2 handles the common failures online without rollback, checkpoint recovery is invoked much less often, and the checkpoint interval can be increased according to the remaining correlated-failure risk, which reduces checkpoint I/O and runtime overhead.

D ADDITIONAL EXPERIMENTS

D.1 NODE-LEVEL FAILURES

Algorithm. A node-level failure is handled by treating all failed ranks on the node as a single failure set F. For each affected EP group (d, p), let $F_{d,p} := \{e : (d, p, e) \in F\}$ be its failed EP ranks and $\Omega_{F_{d,p}} := \bigcup_{e \in F_{d,p}} \Omega_e$ the union of their experts. MeCeFO-v2 distributes $\Omega_{F_{d,p}}$ across the surviving EP peers $[E] \setminus F_{d,p}$, which reside on other nodes, using a balanced map $\varphi_F : \Omega_{F_{d,p}} \to [E] \setminus F_{d,p}$ with $\big||\varphi_F^{-1}(e)| - |\Omega_{F_{d,p}}|/(E - |F_{d,p}|)\big| \leq 1$. The missing parameters and optimizer states are restored from DP replicas on other nodes as described in Appendix C, and the subsequent protection computation is unchanged. For a single failed rank, this construction reduces to the one in Sec. 4.1. Compared with a single-GPU failure, each surviving peer absorbs a larger share of the failed workload, about $|F_{d,p}|/(E - |F_{d,p}|)$ of a shard instead of $1/(E - 1)$, which increases the memory and compute pressure on the protected ranks.

Placement assumption. Supporting node-level failures requires failure-domain-aware placement: each EP group spans multiple nodes, and the DP replicas of the same shard reside on different nodes. A node failure then leaves surviving EP peers in every affected EP group and at least one healthy DP replica of every affected shard. If all replicas of some state are lost, recovery falls back to checkpoints (Appendix C).

Setup. We evaluate physical 8-GPU node failures on our 128-GPU H20 cluster with $D = 2$, $P = 4$, and $E = 16$. Each 16-rank EP group spans two nodes, so taking one node offline removes eight ranks from the same EP group while leaving eight off-node peers to share their workload. The no-fault reference uses the identical placement, model, batch configuration, initialization, and data order. For a fair comparison with the single-GPU setting, we match the average number of unavailable GPUs over time to that in the BASE-FAIL scenario, and we also report single-GPU failures under the same placement.

Results. Table 7 shows that, across the three model scales, node-level failures reduce throughput by only 0.97–2.17% relative to the no-fault run, with a final PPL increase of at most 0.17. This is comparable to single-GPU failures, which reduce throughput by 1.68–2.87% with a PPL increase of at most 0.12. These results show that MeCeFO-v2 remains effective beyond single-GPU failures while preserving the same optimization framework.

Table 8: Downstream evaluation after supervised finetuning on Tulu-3-SFT-Mixture. Higher is better.

Scenario	MMLU	HellaSwag	BBH	ARC-E	ARC-C	IFEval	Avg.
Baseline	**30.55**	64.80	22.85	**78.07**	44.03	16.45	42.79
Sparse-Fail	29.94	64.12	21.67	76.22	45.14	16.27	42.23
Base-Fail	29.85	65.02	**23.70**	77.57	43.94	**18.67**	**43.13**
Frequent-Fail	29.98	63.28	22.57	76.72	**45.64**	16.82	42.50
Slow-Recover	30.08	64.69	22.31	77.48	43.09	15.71	42.23
Fast-Recover	30.14	**66.19**	23.38	77.61	43.60	17.56	43.08

Table 9: Ablation of attention and MoE compression on Qwen3-MoE 16B-A2B. Throughput is measured in token/s with the "k" suffix for thousands; memory is measured in GB.

Method	Micro-batch size 16		Micro-batch size 32	
	Throughput	Memory	Throughput	Memory
MeCeFO-v2 w/o Attn.+MoE Comp.	16.43k	67.19	–	OOM
MeCeFO-v2 w/o Attn. Comp.	21.23k	55.27	–	OOM
MeCeFO-v2 w/o MoE Comp.	19.79k	60.91	–	OOM
MeCeFO-v2	24.16k	50.32	25.76k	84.75
MeCeFO-v2 w/o Fault	23.51k	48.76	25.39k	78.42

D.2 More Downstream Task Evaluation

We further evaluate the robustness of MeCeFO-v2 under supervised finetuning (SFT). Starting from the trained Qwen3-MoE 16B-A2B checkpoints obtained under different failure scenarios, we finetune all models on the Tulu-3-SFT-Mixture dataset (Lambert et al., 2024) using the same training configuration.

Table 8 reports downstream evaluation results after SFT. The average score ranges from 42.23 to 43.13, compared with 42.79 for the no-failure baseline. The best performance is achieved under Base-Fail, while all other scenarios remain within 0.6 points of the baseline. Across individual tasks, performance differences are similarly small, with no consistent degradation under more severe failure conditions.

These results, together with the zero-shot evaluation in the main text, indicate that MeCeFO-v2 preserves downstream transfer quality both during pretraining and after subsequent finetuning.

D.3 Ablation Study

Table 9 ablates the two compression components used by MeCeFO-v2 on the Qwen3-MoE 16B-A2B model under an EP=4, PP=4 setting with one failed GPU, using H20 96GB GPUs. The ablated variants remove both attention and MoE compression, remove only attention compression, or remove only MoE compression, respectively. Without the full compression strategy, the system consumes substantially more memory at micro-batch size 16 and fails to fit the larger micro-batch setting. In particular, all three ablated variants run out of memory when the micro-batch size is increased to 32, whereas the full MeCeFO-v2 remains feasible with 84.75 GB memory. At the same time, the full method preserves throughput close to the no-failure run: it achieves 24.16k versus 23.51k token/s at micro-batch size 16, and 25.76k versus 25.39k token/s at micro-batch size 32. These results show that the proposed attention and MoE compression policies are complementary: together they relieve failure-induced memory pressure while keeping the effective training throughput nearly unchanged under single-GPU failure.

Table 10 further evaluates the budget-aware automatic controller against manually selected approximation budgets. The manual settings sweep different retained-attention and MoE-compression ratios, but none of them simultaneously optimizes gradient fidelity and system efficiency. In contrast, the automatic controller selects an intermediate policy, $\kappa/H = 6/16$ and $\chi/K = 65\%$, that achieves the lowest relative gradient error among all failure-mode configurations. Compared with the best manual

Table 10: Ablation of the automatic parameter controller on Qwen3-MoE 16B-A2B. The controller jointly selects the retained attention-head budget κ/H and the MoE compression budget χ/K. Throughput is measured in token/s with the "k" suffix for thousands; memory is measured in GB.

Setting	κ/H	χ/K	Rel. grad. error	Throughput	Memory
Manual-1	4/16	75%	11.42%	25.03k	85.86
Manual-2	8/16	50%	9.72%	24.71k	86.23
Manual-3	12/16	25%	14.21%	24.16k	88.16
Auto Controller	6/16	65%	8.43%	25.76k	84.75
No Fault	16/16	0%	0.00%	25.39k	78.42

setting, it reduces the relative gradient error from 9.72% to 8.43%, while increasing throughput from 24.71k to 25.76k token/s. It also uses less memory than all manually tuned settings. These results support the design of online constrained parameter selection: adapting the approximation budget to the observed resource and gradient-error profile yields a better trade-off than fixed hand-tuned policies, preserving training accuracy while improving failover throughput.