

FP8-RL: A Practical and Stable Low-Precision Stack for LLM Reinforcement Learning

Zhaopeng Qiu* · Jingqi Zhang* · Shuang Yu* · Junjie Lai · NVIDIA (*equal contribution)

up to **44%**faster rollout at BF16-level accuracy
Qwen3 8B & 30B MoE · 20K-token OSL

nvidia

Technical report
alphaxiv.org/abs/2601.18150

Motivation · Challenges · Solutions

Why FP8 for RL rollout?

- **Rollout dominates.** Generation takes ~80% of synchronous RL iteration time.
- **Long context is memory-bound.** 20K-token responses stress KV-cache capacity and bandwidth, not FLOPs.
- **FP8 attacks both.** Half the bytes moved, ~2× GEMM throughput on Hopper tensor cores.
- **Deployment dividend.** A policy trained with FP8 rollouts is already validated at low precision — paving the way for efficient FP8 production serving.

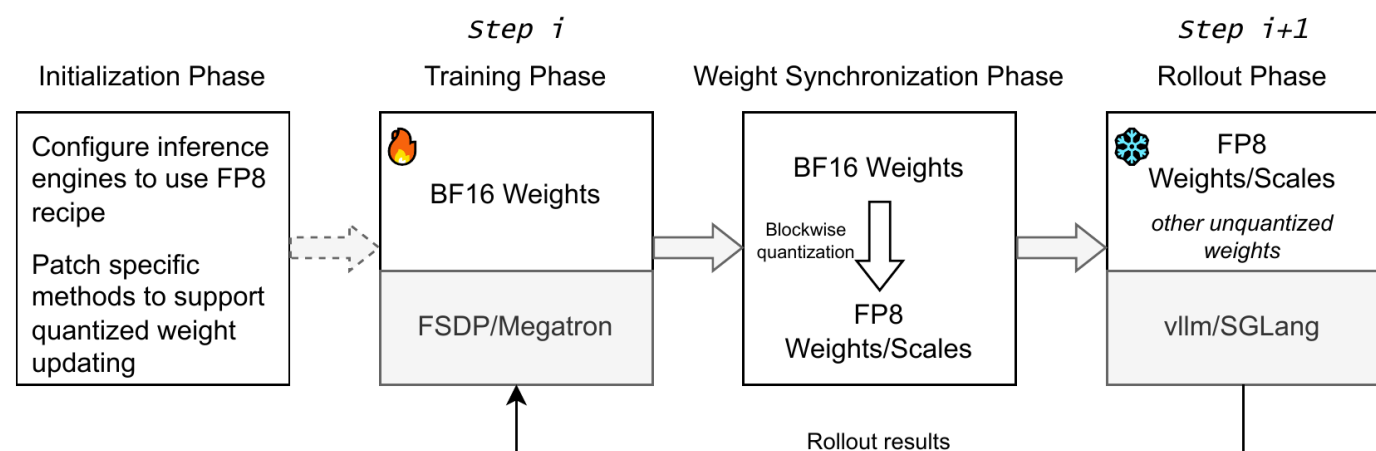
Why it's harder than FP8 inference

- 1 Weights change every step.** The engine must re-quantize and hot-reload the newest policy at step granularity, for every trainer × engine combination.
- 2 FP8 rollout ≠ BF16 trainer policy.** Quantization silently makes on-policy RL off-policy; uncorrected, runs degrade or collapse.

Our answer — four strategies, each paired with token-level TIS:

01 W8A8 linear
rollout only**02** FP8 KV cache
rollout only**03** E2E FP8
rollout + training**04** E2E MXFP8
Blackwell-native

Per-step FP8 weight flow



Every RL step: BF16 weights from FSDP/Megatron → blockwise FP8 → hot-loaded into vLLM/SGLang. Activations are quantized dynamically per forward pass.

Works across the stack

Training	FSDP · Megatron-LM
Inference	vLLM · SGLang
Frameworks	verl · NVIDIA NeMo-RL
FP8 GEMM	DeepGEMM · CUDA 12.9+

Evaluation setup (all experiments)

Models	Qwen3-8B-Base (Dense) · Qwen3-30B-A3B-Base (MoE)
Algorithm	DAPO with online AIME2024 validation
Length	up to 20K-token responses (long-context stress)
Hardware	8×H100 (8B) · 16×H100 (30B) · 32×H100 (E2E)
Backends	vLLM + FSDP · E2E: vLLM 0.12 + Megatron-Core 0.15
Correction	token-level TIS, C = 2, on all FP8 runs

Batching: prompt batch 32 × 16 responses (E2E: 128 × 16); train batch = mini-batch, so each rollout updates the policy exactly once — isolating quantization effects from other off-policy noise.

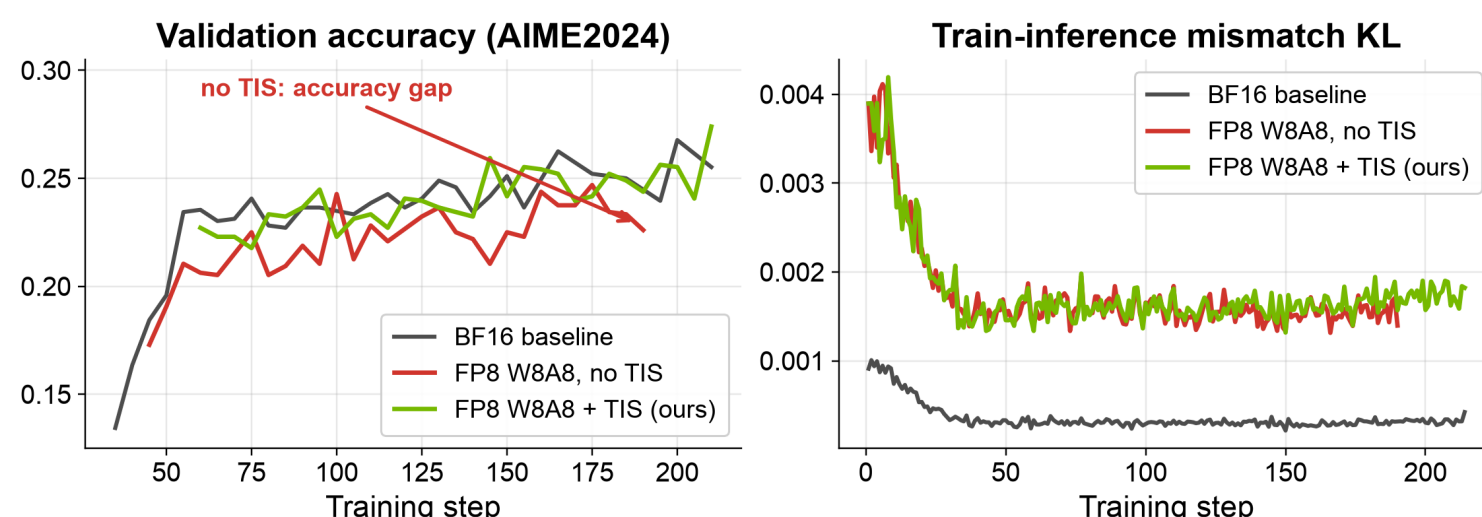
01 W8A8 linear layers — rollout only

$$\hat{W}_{ij} = \text{round}(W_{ij} / s_{ij}) \in \text{E4M3} \quad \cdot \quad \text{one scale per } 128 \times 128 \text{ block}$$

Weights quantized once per weight sync (static); **activations** per forward pass (dynamic).

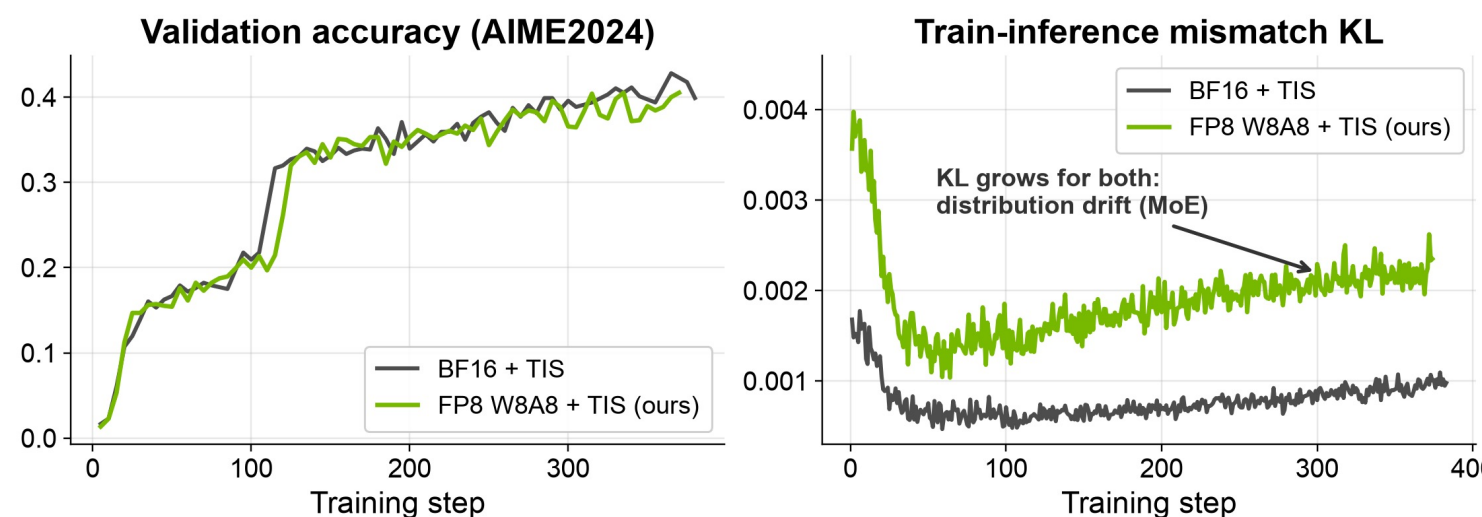
Quantized: QKV/O projections, MLP gate/up/down, MoE experts. **Kept in high precision:** embeddings, norms, lm_head, MoE router.

Dense 8B: TIS is the difference-maker



FP8+TIS tracks BF16 end-to-end. Dropping TIS opens a persistent accuracy gap — at identical train-inference mismatch KL.

MoE 30B: same accuracy, bigger speedup



FP8 W8A8 + TIS matches BF16 + TIS. KL creeps up in both runs — MoE-inherent train-inference distribution drift, not an FP8 artifact.

Rollout speedup (time per token)

10–20%dense Qwen3-8B
8×H100, vLLM**30–50%**MoE Qwen3-30B-A3B
16×H100, vLLM

Gains grow with response length. MoE wins 2–3×: larger GEMMs, half the weight traffic, and freed VRAM that enlarges the KV pool. Keep the MoE router in BF16 — an FP8 router roughly doubles early mismatch KL.

KV FP8 compounds it (8B, 20K OSL). Doubled cache capacity → fewer vLLM preemptions; on the capacity-bound 8B workload, KV-only FP8 outgains linear W8A8 alone.

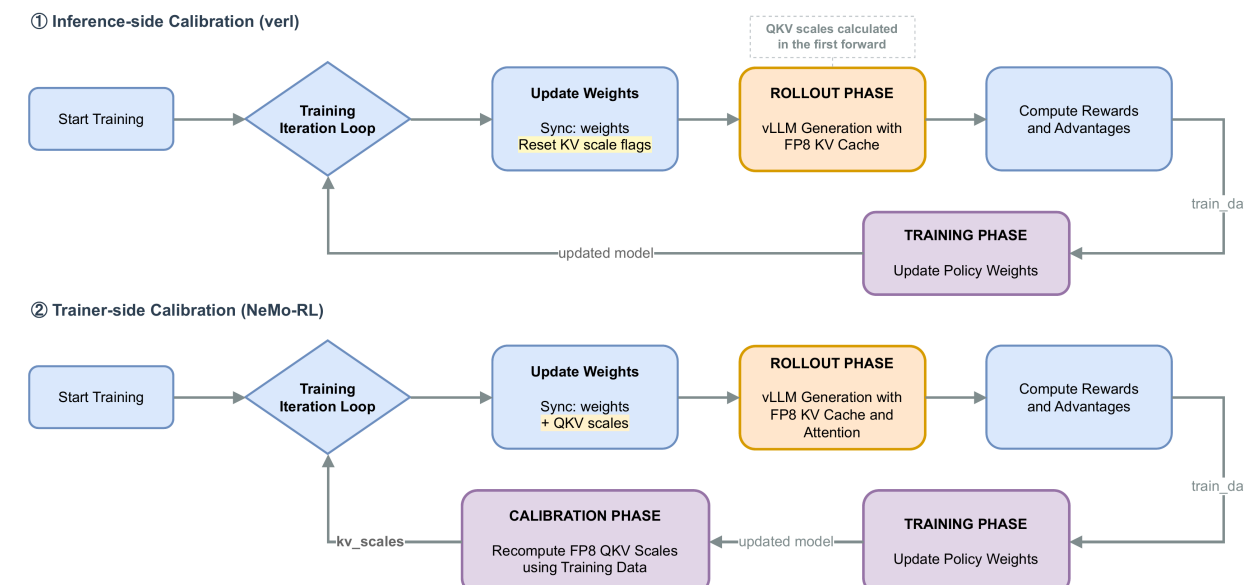
+ Token-level TIS — used by all four strategies

$$w_t^{\text{TIS}} = \min \left(\frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta}^{\text{FP8}}(a_t | s_t)}, C \right), \quad C = 2$$

Reweights policy gradients of FP8-sampled tokens using training-side BF16 log-probs — zero rollout-side overhead.

02 FP8 KV cache — rollout only

Static serving calibrates KV scales once at init. In RL the policy moves every step — stale scales corrupt the cache. Two per-step recalibration designs:



Inference-side calibration

Engine recomputes scales in its first forward pass after each sync (reset calculate_kv_scales). Supports FP8 storage only (attention in BF16) or FP8 storage + FP8 attention.

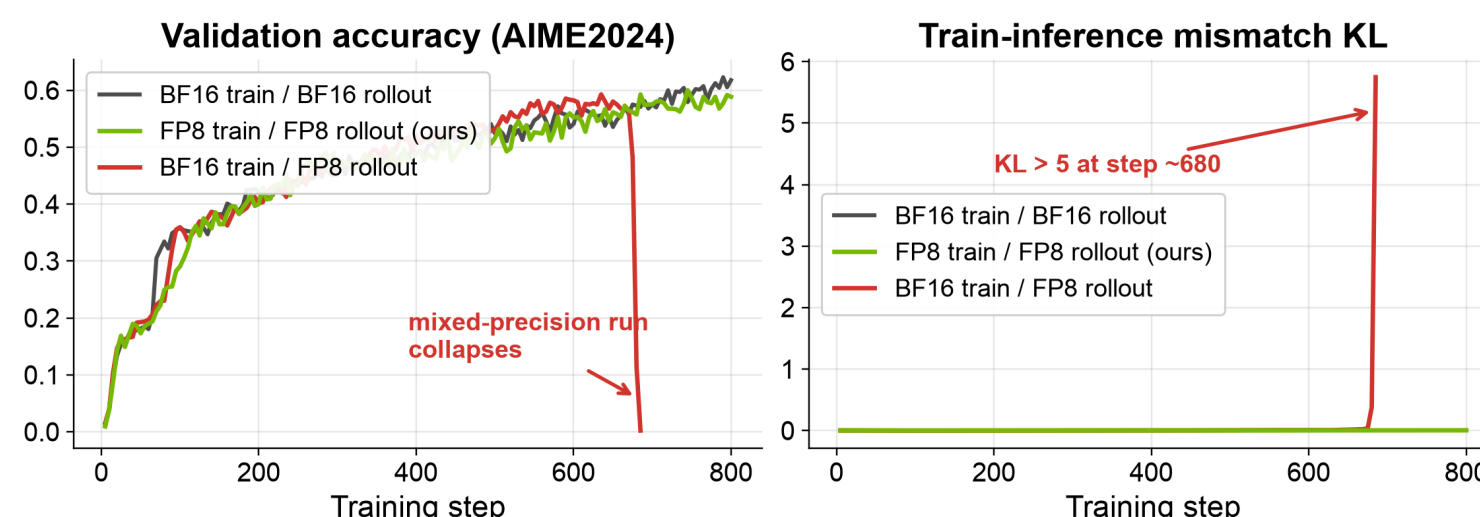
Trainer-side calibration

Adds a calibration phase: recompute scales on fresh training data, ship them with the weight sync (~2–3% step time). Currently supports FP8 storage + FP8 attention.

+38%KV cache FP8 · storage only
attention compute stays BF16**+44%**Full FP8 rollout
W8A8 linear + FP8 KV storage + FP8 attention

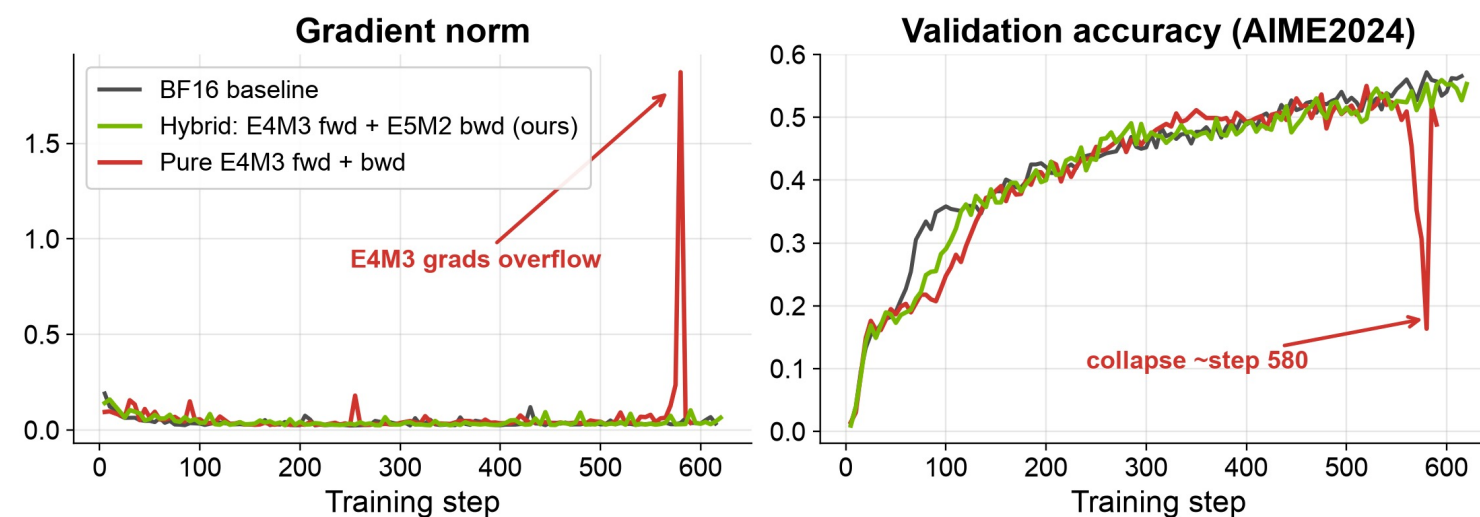
FP8 KV doubles cache capacity → fewer vLLM preemptions, more concurrent sequences. At 20K OSL the 8B workload is capacity-bound, so KV-only FP8 outgains linear W8A8 alone.

03 End-to-End FP8 — rollout + training



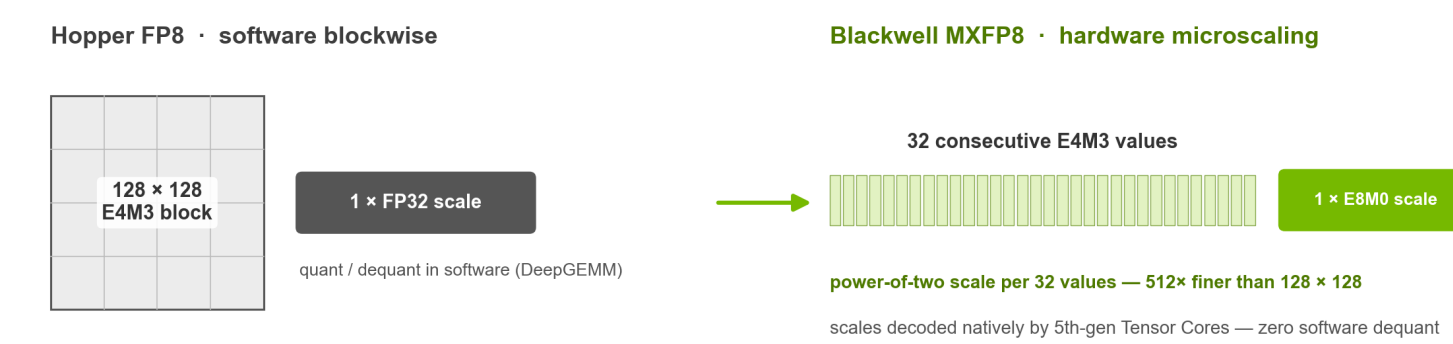
Mixed precision is the unstable point on MoE: BF16-train / FP8-rollout accumulates distribution drift until KL > 5 at step ~680 and collapses — TIS alone is not enough. Applying FP8 W8A8 to linear layers on both sides restores BF16-level stability at ~0.6 on AIME2024.

The backward pass needs E5M2 range



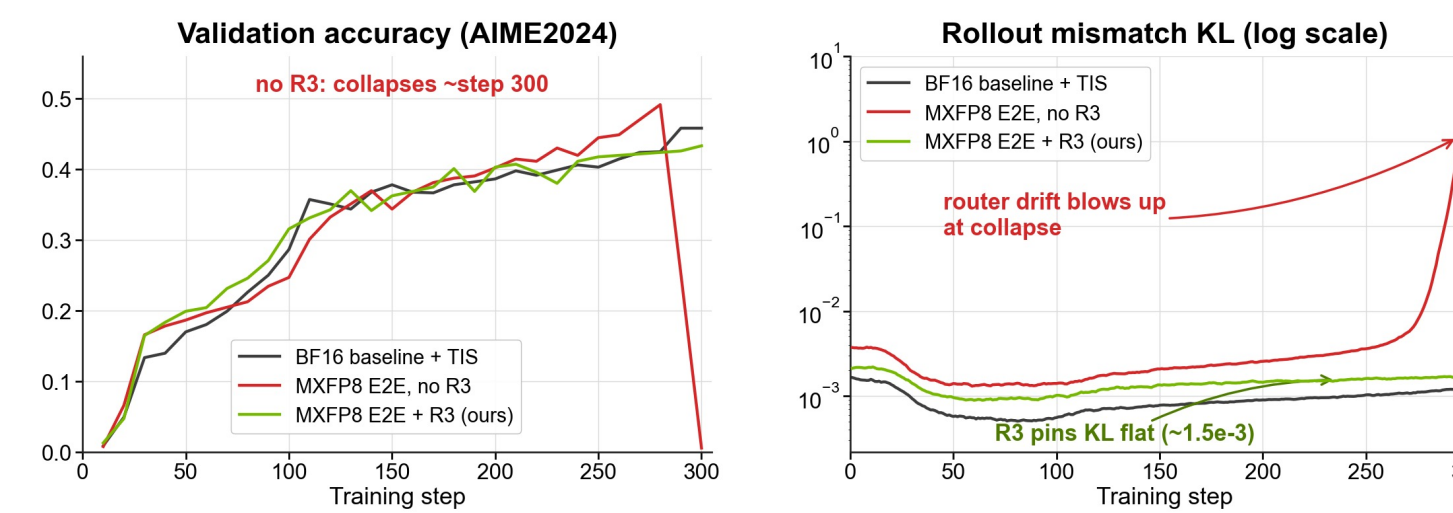
Pure-E4M3 training collapses at ~step 580. Grad-output profiling pins it on MoE fc1: ~5% of tiles exceed E4M3 range on average (layer 0: ~21%); p99 exceedance jumps 26% → 41% in the collapse window.

04 MXFP8 — Blackwell's microscaling FP8 format



- **Microscaling (OCP MX).** Every 32 consecutive values share one E8M0 power-of-two scale; elements stay E4M3 — 512× finer-grained than 128×128 blockwise FP8.
- **Decoded in hardware.** 5th-gen Tensor Cores apply block scales inside the GEMM — no software dequant pass, ~2× the FP8 GEMM rate of Hopper.

End-to-End MXFP8 — mitigating train-inference mismatch

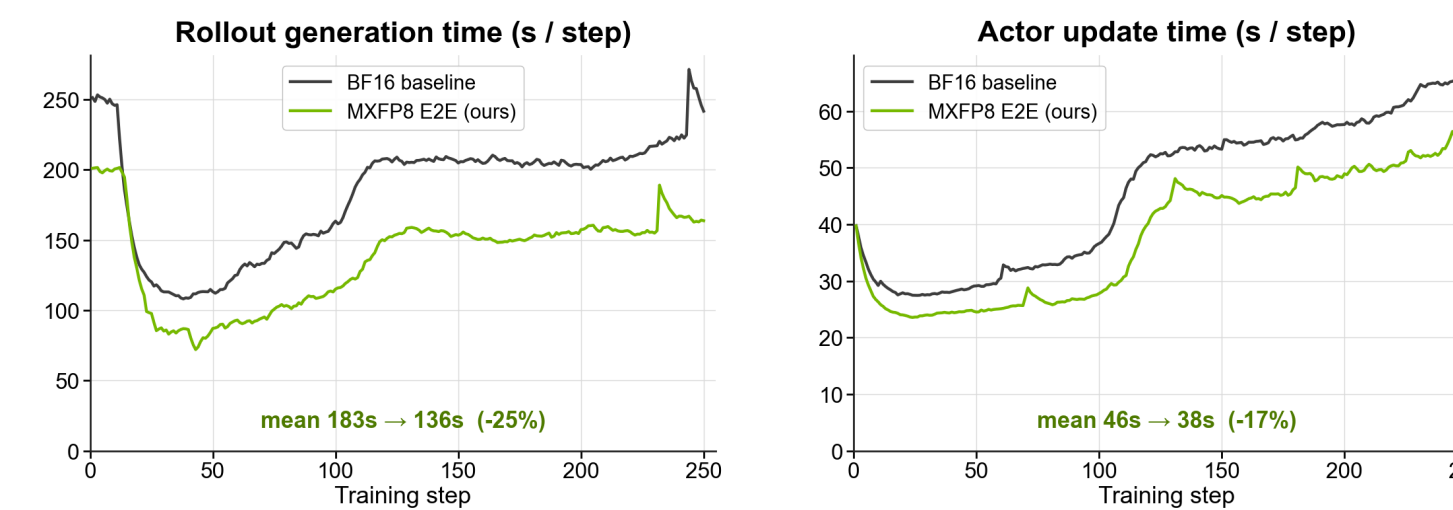


BF16 parity end-to-end. MXFP8 rollout + MXFP8 training tracks the BF16 baseline on AIME2024 — 0.43 vs 0.46 at step 300.

But MoE mismatch grows. Finer-grained quantization amplifies router drift: uncorrected, rollout-vs-trainer KL blows up and accuracy collapses at ~step 300 (red).

Rollout Routing Replay (R3). The trainer replays the expert routing recorded during rollout — KL pinned at ~1.5e-3 and accuracy restored to BF16 level: both the numerics and the outcome recover.

End-to-End speedup



Qwen3-30B-A3B, 20K OSL, first 250 steps (mean/step): rollout generation 183s → 136s (-25%), actor update 46s → 38s (-17%).

Takeaways

- **Up to 44% faster long-context rollout** at BF16-level accuracy — dense and MoE, validated on Hopper.
- **Correction is not optional.** Token-level TIS on every FP8 run; MoE routing under MXFP8 additionally needs R3.
- **E2E FP8 on Hopper buys stability, not trainer speed.** Non-GEMM overhead dominates at 8–30B; the win is lower drift.
- **MXFP8 makes the loop Blackwell-native.** Hardware microscaling end-to-end: -25% rollout / -17% actor-update step time at BF16 parity.
- **Next:** NVFP4 rollout, larger models, multi-turn agentic RL.