PR #2764 完整报告

radixark/miles

perf(megatron): keep policy logits in model precision

合并时间：2026-08-31 10:51

原文链接：http://prhub.com.cn/radixark/miles/pull/2764

执行摘要

- 一句话：policy 前向保留模型精度，按块上转 FP32 省显存
- 推荐动作：值得精读。核心看点是“延迟上转”模式：把整张 logits 的 FP32 物化推迟到 chunk 粒度，配合 Megatron fp32_output 开关实现显存近一个数量级的下降；另一个值得借鉴的细节是把温度缩放放到 FP32 上转之后再做，避免模型精度下的舍入误差，并用 atol=1e-6 的等价性测试固化。fp16 placeholder 的 sum(dtype=torch.float32) 防溢出写法也建议在其它 loss 占位处复用。若团队后续要统一 1F1B 路径，可参考本 PR 的 fp32_output 透传设计。

功能与动机

PR body 明确指出：Float16Module.forward() 默认 fp32_output=True，policy 打分（labels=None）会在 log_probs_chunk_size 生效之前把完整 per-rank [T, V/TP] logits 上转为 FP32。大词表加长序列的 packed 轨迹即使 loss 侧已经按 chunk 处理，也会因整张 FP32 输出而 OOM。因此希望复用现有 chunked log-prob 路径，让每个 chunk 在进入融合 vocab-parallel 交叉熵前再转 FP32，同时为正常 policy-loss 训练前向保留同样的低精度输出，而 value 与其他 loss 维持 FP32 输出。

实现拆解

1. 模型前向输出开关（数据契约变更）：miles/backends/megatron_utils/model.py 的 forward_only() 新增 fp32_output: bool = True 参数并透传到 model(...) 调用，默认值保持旧行为以保护其它调用方；miles/backends/megatron_utils/actor.py 的 MegatronTrainRayActor.compute_log_prob() 显式传 fp32_output=False，让 rollout 打分直接拿到原生 BF16/FP16 logits。训练前向的 forward_step 改为 model(**forward_kwargs, fp32_output=args.loss_type != "policy_loss")：只有 policy loss 走模型精度输出，value/SFT 等仍要求 FP32。配套测试 tests/fast/backends/megatron_utils/test_shared_ppo_lifecycle.py::test_compute_log_prob_keeps_logits_in_model_precision 用 Mock 断言 forward_only 收到 fp32_output is False。
2. 精度转换与温度缩放收敛：miles/backends/training_utils/loss_hub/math_utils.py 新增 _upcast_chunk_to_fp32(logits_chunk, temperature)，用 copy=True 上转 FP32（TP 全词表 CE 前向会原地修改输入），并在 FP32 下完成温度除法；calculate_log_probs_and_entropy() 新增 temperature: float = 1.0 参数，chunk 与非

chunk 路径的 logits.to(torch.float32, copy=True) 全部替换为该辅助函数，entropy 计算也统一走它。true-on-policy 分支新增 assert temperature == 1.0，防止与 _iter_response_chunks 内的缩放形成双重温度。

3. 响应切片路径适配：miles/backends/training_utils/loss_hub/logit_processors.py 的 _iter_response_chunks 把 dtype 断言从 (float32, bfloat16) 扩展为 (float32, bfloat16, float16)；温度除法从该函数中移除（原来在模型精度下执行 logits.div(temperature)），改由 get_log_probs_and_entropy 在调用 calculate_log_probs_and_entropy 时传 temperature=1.0 if args.true_on_policy_mode else args.rollout_temperature，从而把缩放从 BF16 舍入路径移到 FP32 上转之后。
4. FP16 溢出防护：miles/backends/training_utils/loss.py 与 loss_hub/losses.py 中的 autograd 占位表达式由 0 * logits.sum() 改为 0 * logits.sum(dtype=torch.float32)。原因：FP16 logits 求和会溢出为 inf，0 * inf = nan 会毒化梯度图；FP32 sum 保持有限。
5. 测试与回归配套：新增 tests/fast/backends/training_utils/test_chunked_log_probs.py（62 行）覆盖“只有 chunk 上转 FP32”“温度在 FP32 内生效”“FP16 sum 溢出占位”三个契约；test_loss_snapshot.py 新增 grpo_chunked_temp_b2 snapshot 配置（非 true-on-policy + log_probs_chunk_size=8 + rollout_temperature=0.7），并加 module 级 autouse 的单 rank gloo process_group fixture（融合 CE 需要真实 group 对象）；loss_test_utils.make_parallel_state() 在 dist 已初始化时用 dist.group.WORLD 填充 TP group。

关键文件：

- miles/backends/training_utils/loss_hub/math_utils.py（模块 损失计算；类别 source；类型 core-logic；符号 _upcast_chunk_to_fp32, calculate_log_probs_and_entropy）：核心内存优化点：新增 _upcast_chunk_to_fp32 把“按 chunk 上转 FP32 + FP32 温度缩放”收敛为一个辅助函数，calculate_log_probs_and_entropy 新增 temperature 参数并统一走该路径，entropy 计算也随之受益。
- miles/backends/megatron_utils/model.py（模块 模型前向；类别 source；类型 data-contract；符号 forward_only）：数据契约变更入口：forward_only() 新增 fp32_output 参数透传到 Megatron Float16Module；训练前向按 loss_type != "policy_loss" 控制是否输出 FP32，是整条优化链路的源头。
- miles/backends/training_utils/loss_hub/logit_processors.py（模块 响应处理；类别 source；类型 core-logic；符号 _iter_response_chunks, get_log_probs_and_entropy）：把 FP16 纳入 chunked 路径合法输入，并将温度缩放从 _iter_response_chunks 迁移到 calculate_log_probs_and_entropy，保证缩放发生在 FP32 上转之后，是正确性修复的关键一环。
- miles/backends/megatron_utils/actor.py（模块 执行器；类别 source；类型 core-logic；符号 compute_log_prob）：compute_log_prob() 显式传 fp32_output=False，真正让 rollout 打分走模型精度输出；这是用户可感知显存收益的最终落点。
- miles/backends/training_utils/loss.py（模块 损失聚合；类别 source；类型 core-logic；符号 loss_function）：autograd 占位表达式改为 FP32 sum，避免 FP16 logits 求和溢出为 inf 进而产生 0*inf=nan，毒化梯度图。

- miles/backends/training_utils/loss_hub/losses.py（模块 损失函数；类别 source；类型 core-logic；符号 policy_loss_function）：与 loss.py 同源问题：policy_loss_function 的空 log_probs 占位 sum 改为 FP32，保证模型精度输出后仍能正确反传。
- tests/fast/backends/training_utils/test_chunked_log_probs.py（模块 测试；类别 test；类型 test-coverage；符号 test_chunked_log_probs_upcast_only_each_chunk, fake_compute_log_probs, test_temperature_is_applied_in_fp32, naive_compute_log_probs）：新增回归测试套件（62 行），锁定"只有 chunk 上转 FP32""温度在 FP32 内生效""FP16 sum 溢出占位"三个核心契约，是防止后续 dtype 回退的护栏。
- tests/fast/backends/training_utils/loss/test_loss_snapshot.py（模块 测试；类别 test；类型 test-coverage；符号 process_group）：新增 chunked + temperature 的 snapshot 配置与 module 级 process_group fixture，把非 true-on-policy 的融合 CE 路径纳入逐位回归。
- tests/fast/backends/megatron_utils/test_shared_ppo_lifecycle.py（模块 测试；类别 test；类型 test-coverage；符号 test_compute_log_prob_keeps_logits_in_model_precision）：用 Mock 验证 compute_log_prob 调用 forward_only 时一定传 fp32_output=False，锁定 actor 侧接线。
- tests/fast/backends/training_utils/loss/loss_test_utils.py（模块 测试；类别 test；类型 test-coverage；符号 make_parallel_state, _trivial_group）：make_parallel_state 在 dist 已初始化时用真实 dist.group.WORLD 填充 TP group，支撑融合 CE 的集合通信测试。

关键符号：forward_only, compute_log_prob, _upcast_chunk_to_fp32, calculate_log_probs_and_entropy, _iter_response_chunks, get_log_probs_and_entropy, loss_function, policy_loss_function, make_parallel_state, test_chunked_log_probs_upcast_only_each_chunk, test_temperature_is_applied_in_fp32, test_graph_placeholder_sums_in_fp32, test_compute_log_prob_keeps_logits_in_model_precision

关键源码片段

miles/backends/training_utils/loss_hub/math_utils.py

核心内存优化点：新增 _upcast_chunk_to_fp32 把"按 chunk 上转 FP32 + FP32 温度缩放"收敛为一个辅助函数，calculate_log_probs_and_entropy 新增 temperature 参数并统一走该路径，entropy 计算也随之受益。

```
def _upcast_chunk_to_fp32(logits_chunk: torch.Tensor, temperature: float) -> torch.Tensor:
    # copy=True：融合的 TP 全词表交叉熵会在前向中原地修改输入，
    # 拷贝后上转可避免污染模型精度的原始 logits。
    # 温度缩放放在上转之后做，保证除法在 FP32 下进行；
    # 否则在 BF16 下先除温度会引入额外舍入，与 FP32 输入路径不一致。
    chunk = logits_chunk.to(torch.float32, copy=True)
    if temperature > 0 and temperature != 1.0:
        chunk.div_(temperature)
    return chunk

def calculate_log_probs_and_entropy(
```

```
    logits,
    tokens,
    tp_group,
    *,
    chunk_size: int = -1,
    true_on_policy: bool = False,
    sampling_mask=None,
    temperature: float = 1.0,
):
    if true_on_policy:
        # true-on-policy 的缩放发生在 _iter_response_chunks 内部，
        # 这里必须拒绝再缩一次，防止双重温度。
        assert temperature == 1.0, "true-on-policy scales logits in _iter_response_chunks"
        return _calculate_log_probs_and_entropy_true_on_policy(...)

    logits = logits.contiguous()
    if logits.size(0) != 0:
        if chunk_size > 0:
            # 核心内存优化点：整张 logits 保持模型精度分块，
            # 每个 chunk 才上转为 FP32 送给融合 CE，
            # 避免 [T, V/TP] 全量 FP32 的显存峰值。
            num_chunks = (logits.size(0) - 1) // chunk_size + 1
            tokens_chunks = tokens.chunk(num_chunks, dim=0)
            logits_chunks = logits.chunk(num_chunks, dim=0)
            sampling_mask_chunks = (
                sampling_mask.chunk(num_chunks, dim=0)
                if sampling_mask is not None
                else [None] * num_chunks
            )
            log_probs = []
            for tokens_chunk, logits_chunk, sampling_mask_chunk in zip(
                tokens_chunks, logits_chunks, sampling_mask_chunks, strict=True
            ):
                log_prob = compute_log_probs(
                    _upcast_chunk_to_fp32(logits_chunk, temperature),
                    tokens_chunk,
                    tp_group,
                    sampling_mask=sampling_mask_chunk,
                )
                log_probs.append(log_prob)
            log_prob = torch.cat(log_probs, dim=0)
        else:
            # 未开启 chunk 时也统一走 _upcast_chunk_to_fp32，
            # 保证温度缩放在 FP32 下执行，与 chunk 路径语义一致。
            log_prob = compute_log_probs(
                _upcast_chunk_to_fp32(logits, temperature),
                tokens,
                tp_group,
                sampling_mask=sampling_mask,
```

```
            )
    else:
        log_prob = logits.new_zeros((0,))
    return log_prob, entropy
```

miles/backends/megatron_utils/model.py

数据契约变更入口：forward_only() 新增 fp32_output 参数透传到 Megatron Float16Module；训练前向按 loss_type != "policy_loss" 控制是否输出 FP32，是整条优化链路的源头。

```
# forward_only()：Megatron 前向打分入口，新增 fp32_output 透传参数。
# 默认 True 保持既有调用方行为不变；actor 打分时显式传 False，
# 让 Float16Module 直接输出模型精度（BF16/FP16）的 logits，
# 避免在 chunk 化之前物化整张 [T, V/TP] 的 FP32 张量。
@torch.no_grad()
def forward_only(
    f: Callable[..., dict[str, list[torch.Tensor]]],
    args: Namespace,
    model: Sequence[DDP],
    data_iterator: Sequence[DataIterator],
    num_microbatches: Sequence[int],
    rollout_id: int,
    store_prefix: str = "",
    fp32_output: bool = True,
) -> dict[str, list[torch.Tensor]]:
    # ... 省略 batch 读取与数据迭代器重置 ...

    @dumper_phase_util.wrap_forward_step
    def forward_step(data_iterator, model, return_schedule_plan=False):
        # ... 省略 get_batch 与 packed_seq_params ...
        output_tensor = model(
            input_ids=tokens,
            position_ids=None,
            attention_mask=None,
            labels=None,
            packed_seq_params=packed_seq_params,
            loss_mask=batch["full_loss_masks"],
            fp32_output=fp32_output, # 透传模型精度开关到 Float16Module
        )
        return output_tensor, partial(
            f,
            args=args,
            unconcat_tokens=unconcat_tokens,
            total_lengths=total_lengths,
            response_lengths=response_lengths,
            with_entropy=args.use_rollout_entropy,
            max_seq_lens=batch.get("max_seq_lens", None),
        )
    # ... 省略 pipeline 引擎调度 ...
```

```
    # 训练前向的对称处理：只有 policy_loss 走 chunked log-prob 路径，
    # 下游按 chunk 上转 FP32；value / SFT 等其他 loss 仍保留 FP32 输出。
    if (x := batch["multimodal_train_inputs"]) is not None:
        forward_kwargs.update(x)
    output_tensor = model(**forward_kwargs, fp32_output=args.loss_type != "policy_loss")
```

tests/fast/backends/training_utils/test_chunked_log_probs.py

新增回归测试套件（62 行），锁定“只有 chunk 上转 FP32”“温度在 FP32 内生效”“FP16 sum 溢出占位”三个核心契约，是防止后续 dtype 回退的护栏。

```
@pytest.mark.parametrize("model_dtype", MODEL_DTYPES) # 覆盖 BF16 与 FP16
# 验证核心契约：整张 logits 保持模型精度，只有每个 chunk 被上转为 FP32。
def test_chunked_log_probs_upcast_only_each_chunk(monkeypatch, model_dtype):
    kernel_inputs = []

    # 用假 kernel 捕获每次调用时传入 CE 的 shape 与 dtype。
    def fake_compute_log_probs(logits, tokens, _tp_group, *, sampling_mask=None):
        assert sampling_mask is None
        kernel_inputs.append((logits.shape, logits.dtype))
        return torch.zeros((tokens.size(0), 1), dtype=logits.dtype)

    monkeypatch.setattr(math_utils, "compute_log_probs", fake_compute_log_probs)
    logits = torch.zeros((5, 8), dtype=model_dtype)
    tokens = torch.zeros(5, dtype=torch.long)

    math_utils.calculate_log_probs_and_entropy(logits, tokens, None, chunk_size=2)

    # 期望：5 行按 chunk_size=2 切成 [2, 8]、[2, 8]、[1, 8]，
    # 且每个 chunk 进入 kernel 前都已转成 FP32。
    assert kernel_inputs == [
        (torch.Size([2, 8]), torch.float32),
        (torch.Size([2, 8]), torch.float32),
        (torch.Size([1, 8]), torch.float32),
    ]
```

评论区精华

评审侧没有针对 diff 的逐行评论：PR 来自 fork，claude[bot] 提示自动评审关闭，维护者 yueming-yuan 直接批准并点评 “fixes bugs and added UT & snapshot test. approved now”。实际的技术交锋体现在 3 个跟进提交中：1b96101（apply temperature scaling in fp32 inside each logits chunk）把原作者在模型精度下做温度除法的位置改到 FP32 上转之后，并加 assert temperature == 1.0 防止 true-on-policy 双重缩放；94d53ec 补充 chunked policy snapshot 配置；1be6f92 把 FP16 纳入 chunked 路径的合法输入。说明“模型精度输出 + chunked 上转”的主干方案被保留，而精度正确性细节由维护者补齐。

- 温度缩放的精度位置：模型精度除法改为 FP32 内除法 (correctness): 温度缩放统一在 FP32 chunk 内完成；true-on-policy 路径通过 assert temperature == 1.0 保持原有“在响应切片内缩放”的契约，防止双重缩放。

- FP16 logits 受理与 autograd 占位溢出防护 (correctness): 已解决。FP16 输入成为合法路径，占位 sum 强制 FP32 保证 autograd 图有限。
- fork PR 的自动评审关闭与人工批准 (other): 无未解决技术评论；实质技术迭代通过 3 个跟进提交完成。

风险与影响

- 风险：
  1. dtype 契约变更面广：fp32_output 语义影响 forward_only() 的所有调用方与训练前向所有 loss 类型。forward_only 默认 True、训练侧按 loss_type != "policy_loss" 区分，能保住 value/SFT 路径，但任何新增调用方若不显式传参，会静默回退 FP32 输出而丢失内存收益。
  2. 数值精度变化：log-prob 从 BF16 logits 计算，最大绝对误差 9.5367e-7（FP32 基准），训练 loss 与 BF16 梯度 max/mean 绝对差均为 0（损失检查点配置下）。但该验证仅覆盖 TP=1、H200、特定 shape，更大 TP/EP 或更长序列的组合还需持续回归。
  3. 温度缩放语义迁移：原来在 _iter_response_chunks 对整段 logits 做 BF16 除法，现在改为每个 FP32 chunk 内除法。非 true-on-policy 下数值更精确，但若其它模块仍按“模型精度侧已缩放”的假设消费 logits，会出错；true-on-policy 路径靠 assert temperature == 1.0 兜底。
  4. 1F1B 未覆盖：PR body 明确提及 combined 1F1B 的 Megatron PostProcessNode 仍做无条件 float16_to_fp32，该调度下无法享受本优化，后续若要做统一需另起改造。
  5. 新 FP16 输入路径：_iter_response_chunks 现在受理 FP16 logits，任何下游若假设 FP32 或 BF16 可能触发断言或精度问题；logit_processors.py 的 docstring 与断言已同步更新。 - 影响：对用户的直接收益是显存释放：打分阶段每 rank 省约 1.4 GB（1596→156 MiB），训练 forward+backward 每 rank 省约 464 MiB（768→304 MiB），且与 --recompute-loss-function + 正 --log-probs-chunk-size 组合时收益最大（forward 仅 32 MiB）。这让更大词表、更长 packed 轨迹或更大 micro-batch 在同样硬件上可训练。对系统的影响集中在 Megatron 后端的 actor 打分与 policy-loss 训练两条主路径，value/SFT 等其它 loss 行为不变；1F1B 与 true-on-policy 路径保持旧语义。对团队而言，这是一次带有正确性修复的性能优化，测试配套（UT + snapshot）覆盖了新增契约，后续维护者可直接复用 _upcast_chunk_to_fp32 与 snapshot 配置。 - 风险标记：核心训练路径 dtype 契约变更，FP16 输入新路径需回归，1F1B 路径未覆盖，温度缩放语义迁移，依赖逐位精度测试验证

关联脉络

- PR #2818 fix(megatron): keep SFT logits in model precision: 同一功能线的直接延续：把本 PR 的“保持模型精度输出”策略扩展到 SFT logits，改动文件高度重合（model.py、math_utils.py、losses.py、test_loss_snapshot.py）。
- PR #2826 test(loss): CP=2 allgather consistency tests; fix empty-rank log-prob dtype: 继续打磨 chunked log-prob 路径：补齐 CP=2 allgather 一致性测试并修复空 rank 的 log-prob dtype，与本案的 dtype 契约改动同属一条测试主线。

- PR #2200 [RL] Add sampling-support log-prob primitives: 为 math_utils.py 中的 log-prob/entropy 计算管线打下了基础，本 PR 的 _upcast_chunk_to_fp32 正是建立在该管线之上。