PR #2223 完整报告

radixark/miles

Remove --disable-weights-backuper; default eligible colocate launchers to rematerialize

合并时间：2026-08-08 03:24

原文链接：http://prhub.com.cn/radixark/miles/pull/2223

执行摘要

- 一句话：移除过时备份开关，默认启用 rematerialize 权重重建
- 推荐动作：值得精读。该 PR 展示了三个有深度的工程决策：#725 前后内存语义演化如何让一个参数失去存在价值；冻结参数导致分布式 rank 集体序列发散（_ALLGATHER_BASE vs _REDUCE_SCATTER_BASE）的排查与回退；以 --ci-test 自动开启 SHA256 校验作为 CI 安全网的设计。阅读时建议重点关注 tensor_backper.py 的分支收敛和提交历史中默认范围从 6 个加载体收敛到 2 个的过程。

功能与动机

PR body 说明该参数的历史演进：pre-#276 时禁用 backuper 确实省掉一份完整固定宿主拷贝，help 文本也写于那时；#276 为禁用模式添加了 get_cpu_backup 读取路径；#725 将两个备份耦合（disable_param_buffers_cpu_backup = enable_weights_backuper），此后两种模式都只持有恰好一份固定拷贝，参数只剩『无标签机制、通过 TMS 字节恢复』的语义。#1572 提供了真正的零拷贝选项，并在 Qwen3-30B/35B 上通过 SHA256 验证位精确性，因此该参数可以被整体移除，其背后的死代码分支也随之删除。

实现拆解

按 5 个步骤拆解实现过程：

1. 删除启动参数与 Noop 分支：在 miles/utils/arguments.py 中移除 --disable-weights-backuper 的定义；miles/utils/tensor_backper.py 中 TensorBackuper.create 删除 single_tag 参数，不再返回 _TensorBackuperNoop，并删除 _compute_hash_dict / _compute_hash_tensor 两个弱求和哈希辅助函数。这样 enable_weights_backuper 概念彻底消失，backuper 只保留 _TensorBackuperNormal 与 _TensorBackuperMainCast 两条分支。
2. 删除字节恢复读取路径：miles/backends/megatron_utils/update_weight/common.py 中 named_params_and_buffers 与 collect_named_tensors_for_weight_transfer 移除 translate_gpu_to_cpu 参数，删掉 _maybe_get_cpu_backup（它只为 Noop 模式的 TMS 字节恢复服务）；miles/backends/megatron_utils/actor.py 初始化 backuper 时不再传 translate_gpu_to_cpu 与 single_tag。
3. 收紧参数校验：miles/utils/arguments.py 的 _validate_rematerialize_param_from_master_weight 删除 assert args.enable_weights_backuper；miles_validate_args 中

offload_train 分支不再以 enable_weights_backuper 推导 disable_param_buffers_cpu_backup，而是直接置 True；--offload-train-target=disk 的 backuper 相关断言一并移除。注意 head 版本中 _validate_rematerialize_param_from_master_weight 出现了两次调用（功能幂等，但属于合并残留）。

4. 加载体默认开启 rematerialize：scripts/run_glm5_744b_a40b.py、scripts/run_glm5_2_744b_a40b.py 添加 --rematerialize-param-from-master-weight；scripts/run_deepseek.py、scripts/run_glm45_355b_a32b.py 删除 --disable-weights-backuper。提交演进证明 deepseek-v4（mlp.router.weight / tid2eid）与 inkling（rel_proj 为冻结 nn.Parameter）会在 init 时被覆盖断言拒绝，因此默认范围收敛到 glm5 家族；kimi_k25 / glm47-flash（TE FusedAdam 持有 int16 余数 master）、LoRA 加载体和非 colocate 模式保持不开启。
5. 测试与 CI 配套：约 13 个测试文件联动修改。tests/e2e/megatron/model_scripts/test_glm5_744b_a40b_4layer_ci.py、test_deepseek_v4_flash_4layer_ci.py 等去掉 per-test 的 --disable-weights-backuper，改为通过脚本继承默认值；--ci-test 自动开启 SHA256 检查（1fc99a2c），使 GLM/DSA bridge 路径与 deepseek-v4 flash 路径得到位一致性验证；tests/fast/utils/test_arguments.py、test_tensor_backper.py 同步删除对该参数的引用。

关键文件：

- miles/utils/tensor_backper.py（模块 权重备份；类别 source；类型 core-logic；符号 create, _TensorBackuperNoop, init, backup_tags）：核心改动文件：删除 _TensorBackuperNoop、弱哈希辅助函数，TensorBackuper.create 去掉 single_tag 参数，分支收敛为 Normal / MainCast 两种。
- miles/backends/megatron_utils/update_weight/common.py（模块 权重同步；类别 source；类型 core-logic；符号 _maybe_get_cpu_backup）：删除 translate_gpu_to_cpu 参数与 _maybe_get_cpu_backup，权重收集路径不再为 Noop 模式做跨设备翻译。
- miles/utils/arguments.py（模块 参数校验；类别 source；类型 core-logic）：删除 --disable-weights-backuper 参数定义，移除 enable_weights_backuper 相关校验，offload_train 分支直接置 disable_param_buffers_cpu_backup = True。
- miles/backends/megatron_utils/actor.py（模块 Actor 生命周期；类别 source；类型 core-logic）：actor 初始化 backuper 时不再传 translate_gpu_to_cpu 与 single_tag，是调用侧的接入点清理。
- scripts/run_glm5_744b_a40b.py（模块 训练脚本；类别 source；类型 core-logic）：代表加载体默认变更：为 glm5 family 添加 --rematerialize-param-from-master-weight，并让 model_scripts CI 通过脚本继承该默认。
- tests/e2e/megatron/model_scripts/test_glm5_744b_a40b_4layer_ci.py（模块 CI 测试；类别 test；类型 test-coverage）：CI 测试的代表性改动：移除 per-test 的 --disable-weights-backuper，通过脚本默认值继承 rematerialize，并由 --ci-test 自动开启 SHA256 校验。

关键符号：create, _TensorBackuperNoop, named_params_and_buffers, collect_named_tensors_for_weight_transfer, _validate_rematerialize_param_from_master_weight, miles_validate_args

关键源码片段

miles/utils/tensor_backper.py

核心改动文件：删除 _TensorBackuperNoop、弱哈希辅助函数，TensorBackuper.create 去掉 single_tag 参数，分支收敛为 Normal / MainCast 两种。

```
import hashlib
from abc import ABC, abstractmethod
from collections import defaultdict
from collections.abc import Callable, Iterable
from dataclasses import dataclass

import torch

_SourceGetter = Callable[[], Iterable[tuple[str, torch.Tensor]]]


@dataclass(frozen=True)
class MainCastContext:
    # 在训练步结束时，将本 rank 负责的分片从 master weights 写回 params
    cast_main_to_params: Callable[[], None]
    model_chunks: list
    extras_getter: _SourceGetter
    rematerializable_ids: set
    check: bool


class TensorBackuper(ABC):
    @staticmethod
    def create(source_getter, main_cast_ctx: "MainCastContext | None" = None):
        # 曾经的 single_tag 分支在 --disable-weights-backuper 时返回 _TensorBackuperNoop，
        # 只用弱哈希校验、不做真实备份。#725 后两种模式持有相同的一份固定宿主拷贝，
        # 该分支没有内存收益，PR#2223 移除了整个“无标签机制 + TMS 字节恢复”路径。
        if main_cast_ctx is not None:
            return _TensorBackuperMainCast(source_getter=source_getter, ctx=main_cast_ctx)
        return _TensorBackuperNormal(source_getter=source_getter)

    def __init__(self, source_getter: _SourceGetter):
        self._source_getter = source_getter

    @property
    @abstractmethod
    def backup_tags(self):
        raise NotImplementedError

    @abstractmethod
    def get(self, tag: str):
        raise NotImplementedError
```

```
    @abstractmethod
    def backup(self, tag: str):
        raise NotImplementedError

    def copy(self, *, src_tag: str, dst_tag: str):
        raise NotImplementedError

    @abstractmethod
    def restore(self, tag: str):
        raise NotImplementedError


def _hash_tensor_sha256(x: torch.Tensor) -> str:
    """真实（密码学）哈希：与弱求和哈希不同，不匹配必然意味着发生了 bug。"""
    data = x.detach().cpu().contiguous()
    return hashlib.sha256(data.reshape(-1).view(torch.uint8).numpy().tobytes()).hexdigest()
```

miles/backends/megatron_utils/update_weight/common.py

删除 translate_gpu_to_cpu 参数与 _maybe_get_cpu_backup，权重收集路径不再为 Noop 模式做跨设备翻译。

```
def named_params_and_buffers(
    args: Namespace,
    model: Sequence[torch.nn.Module],
    convert_to_global_name: bool = True,
) -> Iterator[tuple[str, torch.Tensor]]:
    # PR#2223 移除 translate_gpu_to_cpu / _maybe_get_cpu_backup 读取路径：
    # 它只为 _TensorBackuperNoop 的“通过 TMS 字节恢复”语义服务，该语义随
    # --disable-weights-backuper 一并删除，backuper 与权重更新不再需要跨设备翻译。
    if convert_to_global_name:
        return _named_params_and_buffers_global(args, model)
    return _named_params_and_buffers_vanilla(model)


def collect_named_tensors_for_weight_transfer(
    args: Namespace,
    model: Sequence[torch.nn.Module],
    convert_to_global_name: bool = True,
    is_expert: bool | None = False,
) -> Iterator[tuple[str, torch.Tensor]]:
    # translate_gpu_to_cpu 参数同步移除，调用方（Megatron actor）不再需要传它
    for name, tensor in named_params_and_buffers(args, model, convert_to_global_name):
        if is_expert is None or is_expert == is_routed_expert_param(name):
            yield name, tensor
```

miles/utils/arguments.py

删除 --disable-weights-backuper 参数定义，移除 enable_weights_backuper 相关校验，offload_train 分支直接置 disable_param_buffers_cpu_backup = True。

```
def _validate_rematerialize_param_from_master_weight(args):
    # 此校验由 miles_validate_args 调用；注意 head 版本中该函数被重复调用了一次（幂等无害）
    if not args.rematerialize_param_from_master_weight:
        return
    if args.debug_train_only:
        # update_weights 从不运行，参数缓冲区不会被暂停，因此静默禁用
        args.rematerialize_param_from_master_weight = False
        return
    assert (
        args.train_backend == "megatron"
    ), "--rematerialize-param-from-master-weight 读取 Megatron 分布式优化器主参数"
    from miles.backends.megatron_utils.lora_utils import is_lora_enabled

    assert not is_lora_enabled(args), "rematerialize 不支持 LoRA"
    assert not args.debug_disable_optimizer, "禁用优化器后没有主参数可重建"
    assert not args.indep_dp, "update_weights 只跑在第一个存活 cell，其他 cell 会一直保留备份"
    assert args.colocate and args.offload_train
    assert args.offload_train_target == "cpu", (
        "磁盘 offload 在 resume 后从 GPU 读回权重，没有可供重建的备份"
    )
    assert args.use_distributed_optimizer
    # 曾断言 args.enable_weights_backuper；PR#2223 删除该 flag 后此约束自然消失
    assert not args.keep_old_actor
    assert not args.use_precision_aware_optimizer or args.optimizer_cpu_offload, (
        "GPU 上 TE FusedAdam 将 master 保存为 int16 余数，没有独立权重可重建"
    )
    assert not args.overlap_param_gather, "重建需要 DDP.start_param_sync，在训练步外运行"
    assert args.compute_advantages_and_returns, "重建在 compute_advantages_and_returns
    块中运行"
    assert args.num_critic_only_steps == 0, "critic-only 步会反复执行 update_weights"
    args.disable_param_buffers_cpu_backup = True
    if args.ci_test:
        # CI 自动开启 SHA256 位一致性校验，测试无需重复传参（与 check_weight_update_equal
        同一模式）
        args.check_rematerialize_param_from_master_weight = True
```

评论区精华

本 PR 无 review 评论（Zhichenzzz 直接 APPROVED），但 33 个 commit 构成一条完整的自我审查链，关键决策记录在提交消息中：

> “Stop backing up frozen params: TMS's default region already restores them”——让 backuper 跳过冻结参数从 TMS 恢复，结果 test_qwen3_5_35b_a3b_lora_ci 挂起：7 个 rank 停在 _ALLGATHER_BASE，3 个停在 _REDUCE_SCATTER_BASE，600s watchdog 中止。

结论是立即回退：get() 对冻结参数的设备与可用性依赖各 rank 的暂停状态，而 LoRA 基础同步会消费该 dict，get() 必须跨 rank 保持一致。

> “inkling: rel_proj is a frozen nn.Parameter … both hit the init-time coverage assert”；deepseek-v4 的 mlp.router.weight / tid2eid 同样被 DDP 跳过。

结论是“config-level eligibility is insufficient—model structure decides”，默认范围缩到有绿色 e2e 证明的 glm5 家族。

> “CI opts into the verification by being CI, not by every test repeating the flag.”——--ci-test 自动开启 SHA256 校验。

- 跳过冻结参数备份导致 LoRA e2e 挂起 (correctness)：立即回退（07ee0092）。原因是 get() 从 TMS 获取冻结参数时，其设备与可用性取决于各 rank 的暂停状态，而 LoRA 基础同步在 skip_base_sync 为 false 时会消费该 dict，get() 必须跨 rank 保持均匀。
- inkling / deepseek-v4 的模型结构资格问题 (design)：默认范围收敛到有绿色 e2e 证明的 glm5 家族；模型结构决定资格，仅凭配置（colocate、optimizer、无 LoRA）不足，必须先跑一次验证运行。
- --ci-test 自动开启 SHA256 校验 (testing)：合并落地：11 个 e2e 测试在无 megatron/sglang 安装的 CPU 套件下全部通过，SHA256 校验覆盖 GLM/DSA bridge 与 deepseek-v4 flash 路径。
- 默认开启 rematerialize 的范围收敛 (design)：只保留 glm5 家族的默认开启；带 CI 覆盖的 deepseek / glm45 加载体移除 --disable-weights-backuper 但暂不默认 rematerialize。

风险与影响

- 风险： 1) CLI 破坏性变更：任何仍传 --disable-weights-backuper 的外部脚本会因未知参数直接失败；仓库内脚本已清理，但用户部署脚本需要同步更新。 2) LoRA colocate 宿主内存回归：冻结基础模型位于默认 TMS 常驻 CPU 备份区域，backuper 的副本成为第二份完整基础模型副本（#725 只对参数缓冲区区域去重）。b396e1f 试图跳过冻结参数但引发 rank 发散，已回退，问题留待专门 PR 解决。 3) 模型资格取决于结构而非配置：inkling 与 deepseek-v4 因冻结参数在 init 时被覆盖断言拒绝，未来若扩展默认范围必须先跑一次验证运行，否则会启动即失败。 4) 代码瑕疵：miles/utils/arguments.py head 中 _validate_rematerialize_param_from_master_weight 被调用两次，功能幂等无害，但应清理。
  - 影响：影响范围覆盖 Megatron 后端 actor 初始化、权重更新读取路径与所有 colocate 启动脚本：glm5 / glm5.2 / deepseek / glm45 / deepseek-v4 / inkling 的 model_scripts CI 均受影响（通过脚本继承默认值）；kimi_k25、glm47-flash、LoRA 与非 colocate 模式保持原状。对用户主要是命令行兼容性破坏与 LoRA colocate 宿主内存回退；对系统而言死代码被清除，参数面更小，rematerialize 成为唯一零拷贝权重重建方案，且 CI 获得了 SHA256 位一致性安全网。团队需要同步示例与文档中对该参数的引用。
  - 风险标记：CLI 参数破坏性删除，LoRA colocate 宿主内存回归，冻结参数导致 rank 发散（已回退），模型资格取决于结构而非配置，重复校验调用（幂等但需清理）

关联脉络

- PR #1572 [optim]--rematerialize-param-from-master-weight: save the bf16 weight backup in colocate: 本 PR 直接堆叠在 #1572 之上；rematerialize 是 #1572 引入的零拷贝方案，#2223 将其设为默认并删除被取代的备份开关。
- PR #2077 Reject --disable-weights-backuper for LoRA + colocate + offload-train: #2077 为 LoRA + colocate + offload 下的 backuper 参数添加了防守断言；本 PR 移除了参数本身，此类断言随之失去存在意义。
- PR #2203 Revert "Reject --disable-weights-backuper for LoRA + colocate + offload-train" (#2077): #2203 回滚了 #2077 的过宽断言，是删除该参数前清理参数面的前奏；本 PR 最终完成参数删除。
- PR #2224 fix: derive --critic-save from --save so PPO critic checkpoints are not silently skipped: 同属参数体系清理：以默认推导取代用户手工维护的标志，减少参数间不一致的空间。
- PR #2226 fix: preserve routing-replay state around MTP spec creation: 同样涉及 megatron actor / model_provider 中权重与生命周期状态的保存恢复，与本 PR 的权重备份语义处于同一区域。