PR #2321 完整报告

THUDM/slime

[cleanup] extract create_weight_updater to make actor's init func cleaner

合并时间：2026-08-24 21:27

原文链接：http://prhub.com.cn/THUDM/slime/pull/2321

执行摘要

- 一句话：新增 create_weight_updater 工厂，精简 actor 初始化
- 推荐动作：值得快速浏览，重点学习两点：一是如何用函数级延迟导入 + monkeypatch sys.modules 注入假模块来零成本测试内部 import 分支；二是把散落在核心初始化函数里的分派逻辑收敛为工厂函数时，如何用断言保住配置约束。若后续需要扩展权重同步通道，此工厂是天然的挂载点。

功能与动机

PR 标题明确说明目的：*extract create_weight_updater to make actor's init func cleaner*。actor.py 的 init 函数中内联了约 29 行基于 update_weight_mode / update_weight_transport / colocate 的分支选择与三个 Updater 的模块级导入，使得初始化流程冗长且难以单独测试。提取工厂后，init 只声明需要权重同步器，分支选择与参数装配被集中封装并可独立验证。

实现拆解

1. 在 slime/backends/megatron_utils/update_weight/init.py 新增工厂函数 create_weight_updater：按 delta → disk → colocate → distributed 顺序选择具体实现类，全部使用函数内延迟导入避免包级循环依赖；构造后统一设置 weight_version = getattr(args, "update_weight_start_version", 0)，与原逻辑完全一致。
2. 修改 slime/backends/megatron_utils/actor.py：删除 UpdateWeightFromDisk / UpdateWeightFromDistributed / UpdateWeightFromTensor 三个模块级导入，把 init 中约 25 行内联分支替换为一次 create_weight_updater(...) 调用；weights_getter 仍绑定 weights_backuper.get("actor")，model_name 与 quantization_config 的推导表达式原样保留，行为不变。
3. 新增 tests/test_update_weight_factory.py：使用 _FakeUpdater 与 monkeypatch.setitem(sys.modules, ...) 拦截工厂函数内的延迟导入，参数化覆盖 delta / disk / colocated / distributed 四种组合，断言选类正确、args / model / weights_getter / model_name / quantization_config 完整透传且 weight_version == 7。
4. 无配置、schema 或部署配套改动；仓库现有 e2e 测试继续作为 actor 集成路径的回归防线。

关键文件：

- slime/backends/megatron_utils/update_weight/__init__.py（模块 权重同步；类别 source；类型 core-logic；符号 create_weight_updater）：新增 create_weight_updater 工厂函数，这是本次重构的核心：统一封装了按 mode / transport / colocate 分发到四种 Updater 实现的逻辑，并统一回填 weight_version。
- slime/backends/megatron_utils/actor.py（模块 训练 Actor；类别 source；类型 dependency-wiring；符号 init, MegatronTrainRayActor）：MegatronTrainRayActor.init 是本重构的调用端：删除了约 29 行内联分支和三个 Updater 导入，替换为工厂调用，使初始化函数显著变短。
- tests/test_update_weight_factory.py（模块 工厂单测；类别 test；类型 test-coverage；符号 _FakeUpdater, test_create_weight_updater_selects_implementation）：新增的参数化单测，用 monkeypatch 注入 sys.modules 拦截工厂内部的延迟导入，验证四种 mode / transport / colocate 组合的选择与参数透传，是本次重构可独立验证的关键配套。

关键符号：create_weight_updater, MegatronTrainRayActor.init, test_create_weight_updater_selects_implementation

关键源码片段

slime/backends/megatron_utils/update_weight/__init__.py

新增 create_weight_updater 工厂函数，这是本次重构的核心：统一封装了按 mode / transport / colocate 分发到四种 Updater 实现的逻辑，并统一回填 weight_version。

```
from __future__ import annotations

from argparse import Namespace
from collections.abc import Callable, Mapping, Sequence
from typing import Any

import torch


def create_weight_updater(
    args: Namespace,
    model: Sequence[torch.nn.Module],
    weights_getter: Callable[[], Mapping[str, torch.Tensor]],
    *,
    model_name: str,
    quantization_config: dict[str, Any] | None,
):
    """Select and construct the weight updater for the configured transport."""
    # 读取配置：mode 决定同步粒度，transport 决定传输通道，colocate 决定是否同卡直传
    update_weight_mode = args.update_weight_mode
    update_weight_transport = args.update_weight_transport

    if update_weight_mode == "delta":
        # delta 模式仅支持 disk 传输：每个 engine 的 /pull_weights 把增量应用到本地 checkpoint，
        # engine 再通过 vanilla update_weights_from_disk 重载，因此不允许 colocate。
```

```
        assert not args.colocate, "--update-weight-mode=delta is not supported with --colocate"
        assert update_weight_transport == "disk", "--update-weight-mode=delta requires --update-
        weight-transport=disk"
        from .update_weight_from_disk_delta import UpdateWeightFromDiskDelta

        update_weight_cls = UpdateWeightFromDiskDelta
    elif update_weight_transport == "disk":
        from .update_weight_from_disk import UpdateWeightFromDisk

        update_weight_cls = UpdateWeightFromDisk
    elif args.colocate:
        from .update_weight_from_tensor import UpdateWeightFromTensor

        update_weight_cls = UpdateWeightFromTensor
    else:
        # 剩余组合约束为 full + nccl + 非 colocate 的分布式同步
        assert update_weight_mode == "full"
        assert (
            update_weight_transport == "nccl"
        ), f"unsupported weight sync mode/transport: {update_weight_mode!r}/{update_weight_
        transport!r}"
        from .update_weight_from_distributed import UpdateWeightFromDistributed

        update_weight_cls = UpdateWeightFromDistributed

    # 统一构造并回填 start_version，行为与原 actor.init 内联逻辑完全一致
    updater = update_weight_cls(
        args,
        model,
        weights_getter,
        model_name=model_name,
        quantization_config=quantization_config,
    )
    updater.weight_version = getattr(args, "update_weight_start_version", 0)
    return updater
```

slime/backends/megatron_utils/actor.py

MegatronTrainRayActor.init 是本重构的调用端：删除了约 29 行内联分支和三个 Updater 导入，替换为工厂调用，使初始化函数显著变短。

```
        # 权重同步器选择逻辑已收敛到 update_weight 包级工厂，
        # init 只负责注入当前 actor 的权重复制源，降低初始化函数复杂度。
        self.weight_updater = create_weight_updater(
            self.args,
            self.model,
            weights_getter=lambda: self.weights_backuper.get("actor"),
            model_name=type(self.hf_config).__name__.lower() if self.args.model_name is None
            else self.args.model_name,
            quantization_config=getattr(self.hf_config, "quantization_config", None),
```

```
        )

        # empty cache after initialization
        clear_memory()
```

tests/test_update_weight_factory.py

新增的参数化单测，用 monkeypatch 注入 sys.modules 拦截工厂内部的延迟导入，验证四种 mode / transport / colocate 组合的选择与参数透传，是本次重构可独立验证的关键配套。

```
import sys
import types
from argparse import Namespace

import pytest

from slime.backends.megatron_utils.update_weight import create_weight_updater

NUM_GPUS = 0


class _FakeUpdater:
    # 与各实现类保持相同签名，方便断言 create_weight_updater 的参数透传
    def __init__(self, args, model, weights_getter, *, model_name, quantization_config):
        self.args = args
        self.model = model
        self.weights_getter = weights_getter
        self.model_name = model_name
        self.quantization_config = quantization_config
        self.weight_version = 0


@pytest.mark.unit
@pytest.mark.parametrize(
    ("mode", "transport", "colocate", "module_name", "class_name"),
    [
        pytest.param("delta", "disk", False, "update_weight_from_disk_delta",
        "UpdateWeightFromDiskDelta", id="delta"),
        pytest.param("full", "disk", False, "update_weight_from_disk", "UpdateWeightFromDisk", id=
        "disk"),
        pytest.param("full", "nccl", True, "update_weight_from_tensor", "UpdateWeightFromTensor",
        id="colocated"),
        pytest.param(
            "full",
            "nccl",
            False,
            "update_weight_from_distributed",
            "UpdateWeightFromDistributed",
            id="distributed",
        ),
```

```
    ],
)
def test_create_weight_updater_selects_implementation(monkeypatch, mode, transport, colocate,
module_name, class_name):
    full_module_name = f"slime.backends.megatron_utils.update_weight.{module_name}"
    fake_module = types.ModuleType(full_module_name)
    setattr(fake_module, class_name, _FakeUpdater)
    # 工厂实现内部使用函数级延迟导入，这里直接向 sys.modules 注入假模块完成拦截
    monkeypatch.setitem(sys.modules, full_module_name, fake_module)

    args = Namespace(
        update_weight_mode=mode,
        update_weight_transport=transport,
        update_weight_start_version=7,
        colocate=colocate,
    )
    model = [object()]

    def weights_getter():
        return {"weight": object()}

    updater = create_weight_updater(
        args,
        model,
        weights_getter,
        model_name="model",
        quantization_config={"quant_method": "test"},
    )

    assert isinstance(updater, _FakeUpdater)
    assert updater.args is args
    assert updater.model is model
    assert updater.weights_getter is weights_getter
    assert updater.model_name == "model"
    assert updater.quantization_config == {"quant_method": "test"}
    assert updater.weight_version == 7
```

评论区精华

本 PR 没有任何 review 评论（comments_count = 0，review_comments_count = 0），由作者 zhuzilin 自审并合并。代码中的关键决策可归纳为两点：一是工厂函数内全部使用延迟导入（函数内 from ... import），既避免包级循环依赖，也让测试能通过向 sys.modules 注入假模块拦截具体实现；二是 delta 模式与 colocate 冲突、非 full/nccl 组合均在工厂入口用 assert 拦截，使非法配置更早暴露。

- 暂无高价值评论线程

风险与影响

- 风险： 1) 行为等价回归风险：actor.init 属于训练 actor 核心启动路径，重构为纯移动但任何参数错位都会导致权重同步异常；新增单测仅覆盖工厂，actor 集成层面仍依赖现有 e2e 用例兜底。 2) 测试技巧耦合：tests/test_update_weight_factory.py 通过 monkeypatch.setitem(sys.modules, ...) 拦截延迟导入，若将来工厂改为模块级导入，该测试会真实加载实现类（可能触发 CUDA 初始化）而失效。 3) 配置合法性断言前移：delta 模式原先在 actor.init 中 assert，现在在工厂入口统一断言，其他调用方（如果有）会得到更早的失败信号；潜在兼容性影响是错误提示的触发时机发生变化，但错误语义一致。
  - 影响：影响局限于 megatron_utils 模块内部：对外 API 无变化，训练 / 推理行为完全一致。对团队而言，权重同步的分发逻辑首次获得独立单元测试，后续新增 transport（如 RDMA）或调整配置校验时不再需要触碰庞大的 actor.init；同时工厂作为 update_weight 包的公共入口，也便于其他模块（如调试工具、离线脚本）复用同样的选择逻辑。
  - 风险标记：核心初始化路径变更，行为等价依赖现有 e2e 覆盖，测试与延迟导入实现耦合

关联脉络

- PR #2322 [cleanup] Remove rollout_validation.py: 同属 cleanup 系列，都在精简初始化相关代码路径，与本 PR 的提取工厂是同方向的代码结构整理。
- PR #2316 Remove megatron_patch for memory optimization: 同为 megatron_utils 适配层的简化清理，与本 PR 共享模块上下文。
- PR #2298 [NFC] Add observability subfolder: 对 megatron_utils 做目录重组，actor.py 被反复触碰，本 PR 进一步收敛其初始化逻辑。
- PR #2271 fix transform_ue8m0 in fp8 convert: 涉及 update_weight 子模块的权重处理与转换逻辑，与本次工厂化共享同一模块边界。