PR #2250 完整报告

THUDM/slime

Add lightweight rollout hooks and sampling controls

合并时间：2026-08-04 10:40

原文链接：http://prhub.com.cn/THUDM/slime/pull/2250

执行摘要

- 一句话：新增 rollout 采样钩子与动态采样回退，保留 DeepEP TMS 状态
- 推荐动作：值得精读：slime/rollout/sample_hooks.py 的 hook 机制设计（签名过滤 + 同步 / 异步统一 + no-op 默认）、slime/backends/megatron_utils/__init__.py 的 TMS 状态 try/finally 恢复模式，以及 should_drop_dynamic_filter_output 的回退阈值判定。建议未来在真实多卡环境验证 TMS patch，并观察动态采样回退对训练数据分布的实际影响。

功能与动机

PR body 明确提出四个目标："Preserve DeepEP TMS state, make router logging configurable, add generic per-sample rollout hooks, and support a one-round dynamic sampling fallback"，并要求补充 stateless Adam 的测试覆盖。背景在于：DeepEP 持久 buffer 可能在 TMS 禁用区间内初始化，原 patch 硬编码"先关后开"在异常路径下无法保证状态恢复；动态采样过滤一旦拒绝过多分组，剩余候选可能不足以填满本轮 batch，需要一次性的单轮回退；各种 per-sample 后处理需求（标记来源、注入评估信息、统计）需要一个统一轻量的扩展点，而不是散落在各 rollout 阶段里的定制代码。

实现拆解

1. 新增通用 per-sample rollout hooks：创建 slime/rollout/sample_hooks.py，提供 apply_rollout_sample_hooks 对 Sample 叶子递归应用 hook，_accepted_kwargs 按 hook 函数签名过滤注入参数（如 rollout_id、evaluation），并统一兼容同步与异步 hook；set_current_rollout_id 通过模块级状态把当前 rollout id 透传给每个样本。随后在 slime/rollout/sglang_rollout.py、slime/ray/rollout.py 中接入调用，slime/utils/arguments.py 注册 rollout_sample_hook_path 配置，slime/utils/misc.py 补充 load_function 相关导入。未配置时直接返回原值，不影响现有流程。
2. 保留 DeepEP TMS 状态：修改 slime/backends/megatron_utils/__init__.py 中包装 deep_ep.Buffer.__init__ 的 new_init，先通过 tms_get_interesting_region() 读取原状态，再临时关闭 TMS 区间，用 try/finally 保证无论初始化成功与否都能恢复原状态；原来的硬编码恢复 True 改为恢复原值，增强可重入性。
3. 单轮动态采样回退：在 slime/rollout/filter_hub/base_types.py 的 DynamicFilterOutput 中新增 keep_when_insufficient 字段，并新增 should_drop_dynamic_filter_output 判定逻辑：当 remaining_batch_size <= target_data_size 时保留本应被拒的分组，避免再启动一轮 oversampling。slime/rollout/filter_hub/dynamic_sampling_filters.py 的

check_reward_nonzero_std_with_fallback 据此调整过滤行为。

4. 配置与日志开关：slime/utils/arguments.py 新增或调整 rollout 采样 hook 路径与 router logging 相关参数（具体注册名在上下文中未完全展开，存在少量不确定性）；slime/utils/misc.py 补齐按路径加载函数所需的导入。
5. 测试配套：新增 tests/test_rollout_sample_hooks.py（验证嵌套形状保持、kwargs 过滤、非法返回类型报错、未配置 no-op）、tests/test_deep_ep_tms_patch.py（用 FakeCdll 覆盖正常与失败路径的 TMS 状态恢复）、tests/test_stateless_adam.py（验证 StatelessAdam 与每步重建的 Adam/AdamW 数值一致且不持久化动量张量）；tests/plugin_contracts/test_plugin_path_loading_contracts.py 补充动态采样回退与严格模式的契约测试。

关键文件：

- slime/rollout/sample_hooks.py（模块 采样钩子；类别 source；类型 core-logic；符号 set_current_rollout_id, _accepted_kwargs, _apply_to_sample, apply_rollout_sample_hooks）：新增的通用 per-sample rollout hooks 机制，是整个 PR 的核心扩展点，定义了签名过滤、同步 / 异步统一、嵌套形状保持与 no-op 默认行为。
- slime/backends/megatron_utils/__init__.py（模块 训练后端；类别 source；类型 core-logic；符号 new_init）：修改 DeepEP Buffer 初始化时的 TMS 开关逻辑，从硬编码恢复改为读取原状态 + try/finally 恢复，是保障训练后端异常安全的关键变更。
- slime/rollout/filter_hub/base_types.py（模块 采样过滤；类别 source；类型 core-logic；符号 DynamicFilterOutput, keep_when_insufficient, should_drop_dynamic_filter_output）：定义动态采样回退的核心数据与判定函数，是单轮 oversampling 回退策略的语义基础。
- slime/rollout/filter_hub/dynamic_sampling_filters.py（模块 采样过滤；类别 source；类型 core-logic；符号 check_reward_nonzero_std_with_fallback）：在 check_reward_nonzero_std_with_fallback 中实际应用回退策略，决定低方差分组是否保留。
- slime/rollout/sglang_rollout.py（模块 采样执行；类别 source；类型 dependency-wiring）：接入 apply_rollout_sample_hooks 的调用点，使 SGLang 执行路径能应用样本级后处理。
- slime/utils/arguments.py（模块 参数配置；类别 source；类型 configuration）：注册 rollout_sample_hook_path 与 router logging 相关配置，是 hooks 与日志开关的参数入口。
- slime/ray/rollout.py（模块 采样执行；类别 source；类型 dependency-wiring）：在 Ray rollout 链路中接入 set_current_rollout_id 与 hooks 相关状态透传。
- tests/test_rollout_sample_hooks.py（模块 单元测试；类别 test；类型 test-coverage；符号 sync_hook, async_hook, invalid_hook, test_rollout_sample_hooks_preserve_nested_shape_and_filter_kwargs）：覆盖 sample hooks 的核心行为：嵌套形状保持、kwargs 过滤、非法返回类型、未配置 no-op。
- tests/test_deep_ep_tms_patch.py（模块 单元测试；类别 test；类型 test-coverage；符号 _load_megatron_utils_init, test_deep_ep_init_restores_original_tms_region, test_deep_ep_init_restores_tms_region_after_failure）：用 FakeCdll 模拟 TMS 实现，覆盖正常初始化与初始化失败时的状态恢复，是异常路径保障的关键验证。

- tests/test_stateless_adam.py（模块 单元测试；类别 test；类型 test-coverage；符号 _run_with_reinitialized_adam, test_stateless_adam_matches_reinitialized_adam_each_step, test_stateless_adam_does_not_persist_moment_tensors）：补充 StatelessAdam 数值一致性验证，确认与每步重建的 Adam/AdamW 行为一致且不持久化状态。

关键符号：set_current_rollout_id, apply_rollout_sample_hooks, _apply_to_sample, should_drop_dynamic_filter_output, check_reward_nonzero_std_with_fallback, new_init

关键源码片段

slime/rollout/sample_hooks.py

新增的通用 per-sample rollout hooks 机制，是整个 PR 的核心扩展点，定义了签名过滤、同步 / 异步统一、嵌套形状保持与 no-op 默认行为。

```
from __future__ import annotations

import inspect
from typing import Any

from slime.utils.misc import load_function
from slime.utils.types import Sample

# 模块级状态，用于把当前 rollout 的 id 透传给每个样本的 hook
_current_rollout_id: int | None = None


def set_current_rollout_id(rollout_id: int | None) -> None:
    global _current_rollout_id
    _current_rollout_id = rollout_id


# 只把 hook 签名里声明的参数传进去，兼容 **kwargs 风格的 hook
def _accepted_kwargs(function, kwargs: dict[str, Any]) -> dict[str, Any]:
    signature = inspect.signature(function)
    if any(parameter.kind == inspect.Parameter.VAR_KEYWORD for parameter in signature.
    parameters.values()):
        return kwargs
    return {key: value for key, value in kwargs.items() if key in signature.parameters}


# 按配置顺序依次执行 hook，支持同步 / 异步，并强制约束返回类型
async def _apply_to_sample(args, sample: Sample, paths: list[str], **kwargs) -> Sample:
    for path in paths:
        hook = load_function(path)
        result = hook(args, sample, **_accepted_kwargs(hook, kwargs))
        if inspect.isawaitable(result):
            result = await result
        if result is not None:
            if not isinstance(result, Sample):
```

```
                raise TypeError(
                    f"Rollout sample hook {path!r} returned {type(result).__name__}, expected Sample
                    or None."
                )
            sample = result
    return sample


# 对 Sample 叶子节点应用 hooks，保持嵌套 list 的原有形状；未配置时直接返回原值
async def apply_rollout_sample_hooks(args, value, **kwargs):
    """Apply configured hooks to every Sample leaf while preserving list shape."""

    paths = getattr(args, "rollout_sample_hook_path", None) or []
    if not paths:
        return value
    kwargs.setdefault("rollout_id", _current_rollout_id)
    if isinstance(value, Sample):
        return await _apply_to_sample(args, value, paths, **kwargs)
    if isinstance(value, list):
        return [await apply_rollout_sample_hooks(args, item, **kwargs) for item in value]
    raise TypeError(f"Rollout sample hooks expected Sample or list, got {type(value).__name__}.")
```

slime/backends/megatron_utils/__init__.py

修改 DeepEP Buffer 初始化时的 TMS 开关逻辑，从硬编码恢复改为读取原状态 + try/finally 恢复，是保障训练后端异常安全的关键变更。

```
# 包装 DeepEP Buffer.__init__：临时关闭 TMS 的 interesting region，
# 并在任何路径下恢复原始状态，避免 DeepEP 持久 buffer 被系统误纳入 TMS 追踪。
old_init = deep_ep.Buffer.__init__


def new_init(self, *args, **kwargs):
    tms_impl = torch_memory_saver._impl
    # 没有 TMS 实现时保持原逻辑，不做额外开关操作
    if tms_impl is None:
        return old_init(self, *args, **kwargs)

    cdll = tms_impl._binary_wrapper.cdll
    # 先读取当前状态，而不是假定为 True，保证可重入与可恢复
    original_interesting_region = cdll.tms_get_interesting_region()
    cdll.tms_set_interesting_region(False)
    try:
        old_init(self, *args, **kwargs)
        # DeepEP 持有持久 buffer，可能在自己的内部流上初始化。
        # 在恢复分配追踪之前先同步，确保这些 buffer 的生存期不依赖 TMS 禁用区间。
        torch.cuda.synchronize()
    finally:
        # 即使 old_init 抛异常，也要恢复原来的 TMS 状态
        cdll.tms_set_interesting_region(original_interesting_region)
```

```
deep_ep.Buffer.__init__ = new_init
```

slime/rollout/filter_hub/base_types.py

定义动态采样回退的核心数据与判定函数，是单轮 oversampling 回退策略的语义基础。

```
class DynamicFilterOutput:
    keep: bool
    reason: str | None = None
    # 当拒绝该组后剩余候选不足以填满 rollout batch 时，保留它以避免
    # 再启动一轮 oversampling，这就是“单轮动态采样回退”的核心开关。
    keep_when_insufficient: bool = False


def should_drop_dynamic_filter_output(
    output: DynamicFilterOutput,
    *,
    remaining_batch_size: int,
    target_data_size: int,
) -> bool:
    # 显式保留：不丢弃
    if output.keep:
        return False
    # 允许回退且剩余数量不足：不丢弃，凑足本轮 batch
    if output.keep_when_insufficient and remaining_batch_size <= target_data_size:
        return False
    return True
```

评论区精华

该 PR 无公开 review 评论与讨论线程，由作者 zhuzilin 直接合入；以下分析基于 patch 与测试推断设计意图。值得注意的设计取舍包括：hook 返回值仅允许 Sample 或 None，以此约束扩展点行为；TMS 状态恢复采用读原值 + try/finally，而非硬编码恢复 True。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. rollout 主路径新增调用点：slime/rollout/sglang_rollout.py 与 slime/ray/rollout.py 接入 sample hooks 后，任何 hook 抛异常都会中断整批 rollout；apply_rollout_sample_hooks 对嵌套 list 的递归展开在超大 batch 下可能带来额外 Python 层开销。
  2. DeepEP TMS 兼容性：slime/backends/megatron_utils/__init__.py 新增对 tms_get_interesting_region 的依赖，若 DeepEP 或 torch_memory_saver 版本变化导致接口缺失，import 阶段即可能失败；测试用 FakeCdll 模拟了正常与异常路径，但未覆盖多卡真实环境。

3. 采样数据分布变化：keep_when_insufficient 会保留原本被拒的低质量分组，可能改变训练数据分布，需关注 reward 方差与过拟合风险；should_drop_dynamic_filter_output 的阈值语义（remaining_batch_size <= target_data_size）若理解偏差，可能过度保留。
4. 配置兼容性：slime/utils/arguments.py 新增配置项属于默认关闭能力，对现有用户影响小，但参数名与校验逻辑的跨版本兼容性需要跟踪。 - 影响：对用户：可通过 rollout_sample_hook_path 等配置启用 sample hooks 与单轮动态采样回退，默认行为不变，属于向后兼容的能力扩展。对系统：影响 rollout 数据后处理链路、Megatron 训练后端的 DeepEP 初始化、以及动态采样过滤策略。对团队：sample_hooks.py 提供了一个统一的 per-sample 扩展点，可服务于后续 disaggregated rollout 的样本后处理；StatelessAdam 测试补齐了数值一致性与状态透明性的行为契约。 - 风险标记：核心路径变更，异常路径兜底，采样行为可变，配置兼容性

关联脉络

- PR #2184 sync source_names: 同为 rollout 样本链路增强，都改动 slime/ray/rollout.py 并引入元数据传递，与本次 sample hooks 的 rollout_id 透传互为补充。
- PR #2167 Always requires rollout_top_p_token_ids when rollout_top_p is not 1.0: 同属 rollout 采样控制与校验改进，同样修改 slime/ray/rollout.py，体现采样配置边界条件的持续收口。
- PR #2181 [3/n] Disaggregated rollout: engine-side /pull_weights: rollout 架构演进的一环，sample hooks 机制可作为 disaggregated 模式下采样结果后处理的统一扩展点。
- PR #2208 Support reloading the default process group: 涉及 Megatron 后端初始化与异常恢复路径，与本 PR DeepEP TMS patch 同处 slime/backends/megatron_utils 风险面，可对照其可靠性设计。