PR #6661 完整报告

verl-project/verl

[rollout, vllm] fix: preserve MTP drafter weights during hybrid sleep

合并时间：2026-06-14 16:19

原文链接：http://prhub.com.cn/verl-project/verl/pull/6661

执行摘要

- 一句话：修复混合睡眠中 MTP drafter 权重丢失问题
- 推荐动作：值得阅读的核心是 _sleep_hybrid 中的配置门控条件处理方式：如何安全地从配置对象中获取可选子配置并定义门控变量。此外，测试设计中用 Mock 引擎模拟分布式环境技巧也可借鉴。建议团队成员关注此类 feature-gated 修复的防御性编程方法。

功能与动机

当前混合睡眠路径对所有 full-weight rollout 使用睡眠等级 2。对于 MTP，drafter 拥有 vLLM 初始化的 drafter-only 权重；等级 2 睡眠丢弃模型权重后，后续 actor 权重同步不能保证恢复所有 drafter-only 状态。在 colocated train/rollout 作业且 free_cache_engine=True 时，这会导致 MTP 产生 draft tokens 但零接受。PR #6661 修复此问题。

实现拆解

1. 修改 _sleep_hybrid 方法（vllm_async_server.py）：在原有的 sleep_level 选择逻辑前，先从 self.config 中获取 mtp 对象，并判断 mtp.enable 与 mtp.enable_rollout 是否同时为 True。若为 True，则 sleep_level = 1（保留 drafter 权重），否则沿用原有逻辑（LoRA/NPU 用 level 1，否则 level 2）。
2. 新增 test_mtp_hybrid_sleep_acceptance_on_cpu.py：使用 _FakeMtpEngine 模拟 MTP 引擎，在 CPU 上运行 _sleep_hybrid，验证 sleep 后 drafter 仍然可用，通过 generate_spec_decode_stats 返回非零接受，确保 spec_accept_rate > 0。
3. 新增 test_spec_decode_metrics_on_cpu.py：直接测试 compute_spec_decode_metrics 函数，覆盖三种场景：全零接受、非零接受恢复、带 non_padding_mask 的占位符丢弃。确保指标计算正确。

关键文件：

- verl/workers/rollout/vllm_rollout/vllm_async_server.py（模块 rollout；类别 source；类型 core-logic；符号 _sleep_hybrid）：核心修复文件，修改了 _sleep_hybrid 方法，添加 MTP 配置检测并决定睡眠等级。
- tests/workers/rollout/rollout_vllm/test_mtp_hybrid_sleep_acceptance_on_cpu.py（模块 测试；类别 test；类型 test-coverage；符号 _FakeMtpEngine, init, sleep, reset_encoder_cache）：新增测试，使用 _FakeMtpEngine 模拟 vLLM 引擎，验证 MTP 混合睡眠后 drafter 仍可用且 spec_accept_rate 非零。

- tests/trainer/ppo/test_spec_decode_metrics_on_cpu.py（模块 测试；类别 test；类型 test-coverage；符号 test_spec_decode_metrics_detect_drafts_with_zero_acceptance, test_spec_decode_metrics_report_nonzero_acceptance_after_recovery, test_spec_decode_metrics_drop_padded_placeholders）：新增测试，验证 compute_spec_decode_metrics 函数在零接受、非零接受和带 padding mask 场景下的行为，为 MTP 指标提供基准。

关键符号：_sleep_hybrid, test_mtp_hybrid_sleep_keeps_drafter_available_for_nonzero_acceptance, test_spec_decode_metrics_detect_drafts_with_zero_acceptance, test_spec_decode_metrics_report_nonzero_acceptance_after_recovery, test_spec_decode_metrics_drop_padded_placeholders

关键源码片段

verl/workers/rollout/vllm_rollout/vllm_async_server.py

核心修复文件，修改了 _sleep_hybrid 方法，添加 MTP 配置检测并决定睡眠等级。

```
# verl/workers/rollout/vllm_rollout/vllm_async_server.py

async def _sleep_hybrid(self):
    """HYBRID sleep: adapters and MTP need level=1; full weights need level=2.

    Uses engine.sleep() instead of engine.collective_rpc("sleep") to ensure
    that sleep is properly propagated to all data-parallel worker processes.
    collective_rpc only reaches the TP workers within a single DP shard,
    leaving other DP shards' weights unreleased, which causes OOM during
    FSDP training backward when DP > 1.
    """
    # 从配置中安全获取 mtp 子配置，默认 None
    mtp_config = getattr(self.config, "mtp", None)
    # 仅当 mtp.enable 和 mtp.enable_rollout 同时为 True 时启用 MTP rollout
    mtp_rollout_enabled = (
        mtp_config is not None
        and getattr(mtp_config, "enable", False)
        and getattr(mtp_config, "enable_rollout", False)
    )
    # MTP drafter-only 权重由 vLLM 在初始化时加载，
    # 如果使用 level 2 睡眠（完全丢弃权重），后续 actor 权重同步无法恢复 drafter 状态。
    # 因此 MTP rollout 时强制使用 level 1，保留 drafter 权重。
    # 此外，LoRA 作为适配器只需要 level 1；NPU 当前不支持 sleep_level（可能影响精度）。
    if mtp_rollout_enabled or self.lora_as_adapter or is_torch_npu_available(check_device=False):
        sleep_level = 1
    else:
        sleep_level = 2
    await self.engine.sleep(level=sleep_level)
    if _VLLM_VERSION >= version.parse("0.17.0"):
        await self.engine.reset_encoder_cache()
```

tests/workers/rollout/rollout_vllm/test_mtp_hybrid_sleep_acceptance_on_cpu.py

新增测试，使用 _FakeMtpEngine 模拟 vLLM 引擎，验证 MTP 混合睡眠后 drafter 仍可用且 spec_accept_rate 非零。

```
# tests/workers/rollout/rollout_vllm/test_mtp_hybrid_sleep_acceptance_on_cpu.py

import asyncio
from types import SimpleNamespace
import numpy as np
import pytest

pytest.importorskip("ray")
pytest.importorskip("vllm")

from verl.trainer.ppo.ray_trainer import compute_spec_decode_metrics
from verl.workers.rollout.vllm_rollout import vllm_async_server

class _FakeMtpEngine:
    """最小化模拟 MTP 引擎在混合睡眠中的故障模式。"""

    def __init__(self):
        self.mtp_drafter_available = True # 初始时 drafter 可用
        self.sleep_levels_that_discard_mtp_drafter = {2} # 等级 2 会丢弃 drafter

    async def sleep(self, level: int):
        # 如果睡眠等级会丢弃 drafter，则置为不可用
        if level in self.sleep_levels_that_discard_mtp_drafter:
            self.mtp_drafter_available = False

    async def reset_encoder_cache(self):
        pass

    def sync_actor_weights(self):
        pass

    def generate_spec_decode_stats(self):
        """根据 drafter 是否可用返回不同的接受统计。"""
        num_draft_tokens = 3
        # 如果 drafter 不可用，接受 token 数为 0；否则全部接受
        num_accepted_tokens = num_draft_tokens if self.mtp_drafter_available else 0
        num_verify_steps = 1
        return num_draft_tokens, num_accepted_tokens, num_verify_steps


def test_mtp_hybrid_sleep_keeps_drafter_available_for_nonzero_acceptance(monkeypatch):
    """验证 _sleep_hybrid 在 MTP 启用时不使用等级 2 睡眠，
    从而保留 drafter 可用，最终 spec_accept_rate > 0。"""
```

```
monkeypatch.setattr(vllm_async_server, "is_torch_npu_available",
                    lambda check_device=False: False)

server = object.__new__(vllm_async_server.vLLMHttpServer)
# 配置为 MTP rollout 启用
server.config = SimpleNamespace(mtp=SimpleNamespace(enable=True, enable_rollout=True))
server.model_config = SimpleNamespace(lora_rank=0, lora={})
server.engine = _FakeMtpEngine()

asyncio.run(server._sleep_hybrid())
server.engine.sync_actor_weights()
drafts, accepts, verifies = server.engine.generate_spec_decode_stats()
metrics = compute_spec_decode_metrics(
    spec_drafts=np.array([drafts]),
    spec_accepts=np.array([accepts]),
    spec_verifies=np.array([verifies]),
)

# 断言 spec_accept_rate 和 spec_accept_length 均为正
assert metrics["rollout/spec_accept_rate"] > 0.0
assert metrics["rollout/spec_accept_length"] > 1.0
```

评论区精华

Review 中仅有一条实质讨论：Luosuu 在 vllm_async_server.py 第 955 行请求添加测试。作者 sunnweiwei 回应已在 e46d2574 提交中添加两个测试文件，分别模拟 MTP 混合睡眠故障模式和验证接受率指标。随后 Luosuu 批准（LGTM）。

- 需要为 MTP 混合睡眠添加测试 (testing): 作者添加了测试并通过 CI，PR 被批准合并。

风险与影响

- 风险：
  1. 配置依赖风险：代码通过 getattr(self.config, "mtp", None) 获取 MTP 配置，若 self.config 无 mtp 属性则安全返回 None。但若未来重构中 self.config 的类型或属性结构变更，需保持兼容。
  2. 内存权衡：使用睡眠等级 1 保留了 drafter 权重和更多 GPU 内存，可能导致显存占用比之前的等级 2 更高，但这是正确性前提下的必要代价。
  3. 回归风险：非 MTP 路径行为不变（LoRA/NPU 用 level 1，否则 level 2），已有测试覆盖。
  4. 测试可靠性：新增测试在 CPU 上通过 mock 验证，覆盖了关键路径，但未覆盖真实多 GPU 场景（因硬件限制），不过已在 8xH100 上手动验证。 - 影响：用户影响：对启用 vLLM MTP rollout（mtp.enable=True + mtp.enable_rollout=True）且使用混合睡眠（free_cache_engine=True）的训练任务，修复了 MTP 零接受问题，恢复预期的 speculation acceptance rate。非 MTP 用户无影响，无需修改配置。系统影响：无新增依赖，仅一行核心逻辑变化。团队影响：新增两个单元测试文件，提高模块可测试性，降

低回归风险。

- 风险标记：核心路径变更，配置门控条件，内存权衡

关联脉络

- 暂无明显关联 PR