PR #6456 完整报告

verl-project/verl

[rollout, vllm] fix: use engine.sleep() instead of collective_rpc

合并时间：2026-05-27 14:10

原文链接：http://prhub.com.cn/verl-project/verl/pull/6456

执行摘要

- 一句话：修复 HYBRID 模式下 DP > 1 时 CUDA OOM
- 推荐动作：建议精读: 这是一个典型的"单行修复背后有深度 root cause 分析"的 PR。值得关注的点：
  1. engine.sleep() 与 collective_rpc("sleep") 在 vLLM 分布式架构中的语义差异（DP 协调器 vs 仅 TP 工人）。
  2. 通过消除间接层（collective_rpc）直接调用 actor 方法的重构技巧。
  3. 死代码识别与清理的决策过程。

对于 reviewer，重点关注是否仍有其他路径（如非 HYBRID 模式）意外使用了 collective_rpc 进行 sleep/wake_up。

功能与动机

训练 20B 稠密模型， data_parallel_size=8, tensor_model_parallel_size=1, rollout_mode=HYBRID 时，第一个训练步骤反向传播出现 CUDA OOM。根本原因是 _sleep_hybrid() 中 engine.collective_rpc('sleep') 仅到达单个 DP shard 内的 TP 工人，其他 DP shard 的 ~40GB 模型权重残留 GPU，导致 FSDP 训练时显存耗尽。

实现拆解

1. 替换 core sleep/wake_up 路径

- 文件: verl/workers/rollout/vllm_rollout/vllm_async_server.py
- 变更:
  - _sleep_hybrid() 中将 await self.engine.collective_rpc("sleep", kwargs={"level": sleep_level}) 替换为 await self.engine.sleep(level=sleep_level)。
  - wake_up(self, tags) 中为 HYBRID 模式新增处理：调用 self.engine.wake_up(tags=tags or ...) 并重置前缀缓存，此前该模式直接抛出 ValueError。
- 原因: engine.sleep() 通过 DPAsyncLLM 协调器广播到所有 EngineCore 进程（每个管理一个 DP shard），而 collective_rpc 仅限当前进程的 TP 工人。

2. 消除 ServerAdapter 层的 collective_rpc 绕路

- 文件: verl/workers/rollout/vllm_rollout/vllm_rollout.py
- 变更:
  - 新增 _ensure_server_handle() 辅助方法，封装 lazy-init 逻辑并返回布尔值指示是否应继续。
  - resume() 中从 await self._execute_method("wake_up", kwargs={"tags": tags}) 改为 await self.server_handle.wake_up.remote(tags=tags)。
  - release() 中从 await self._execute_method("sleep", kwargs={"level": ...}) 改为 await self.server_handle.sleep.remote()。
- 原因: _execute_method 内部的 collective_rpc 是 DP 传播问题的根源，直接调用 actor 方法 wake_up.remote()/sleep.remote() 可绕过该问题。同时重用了 _ensure_server_handle 的 lazy-init 逻辑。

3. 清理死代码与无关守卫

- 文件: verl/workers/rollout/vllm_rollout/vllm_async_server.py
  - 移除 collective_rpc() 中无用的 if not hasattr(self, "engine") 守卫（经审查为死代码）。
  - 清理后的逻辑更简洁，消除了潜在的维护陷阱。

4. 测试配套

- 未新增专门的多 GPU 测试（需要 8+ GPU 硬件），但现有 DP=1 的 e2e 测试覆盖了 sleep/wake_up 路径，变更已通过作者在 8×H100 环境上的手动验证。

关键文件:

- verl/workers/rollout/vllm_rollout/vllm_rollout.py（模块 rollout 适配器；类别 source；类型 core-logic；符号 _ensure_server_handle）：ServerAdapter 层，重构了 resume/release 的调用方式，消除了 collective_rpc 绕路；新增 _ensure_server_handle 辅助方法。
- verl/workers/rollout/vllm_rollout/vllm_async_server.py（模块 vLLM 服务端；类别 source；类型 core-logic；符号 wake_up）：vLLM 服务端，修复了 _sleep_hybrid 和 wake_up，替换 collective_rpc 为 engine.sleep/wake_up，支持 HYBRID 模式下的 wake_up，清理了 dead code。

关键符号：_ensure_server_handle, resume, release, wake_up, _sleep_hybrid

关键源码片段

verl/workers/rollout/vllm_rollout/vllm_rollout.py

ServerAdapter 层，重构了 resume/release 的调用方式，消除了 collective_rpc 绕路；新增 _ensure_server_handle 辅助方法。

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

# 新增 _ensure_server_handle 辅助方法，统一 lazy-init 逻辑
# 返回 False 表示非 master 节点，应跳过处理（不再使用 rollout_rank 判断）
def _ensure_server_handle(self) -> bool:
```

```
        """Lazy -init server handle. Returns False if this rank should not proceed."""
        if self.rollout_rank != 0:
            return False
        # Lazy init http server adapter because http server is launched after hybrid engine.
        if self.server_handle is None:
            prefix = self._get_server_name_prefix()
            self.server_handle = ray.get_actor(f"{prefix}server_{self.replica_rank}_{self.node_rank}")
        return True

    async def resume(self, tags: list[str]):
        """Resume rollout weights or kv cache in GPU memory."""
        # 直接调用 server_handle.wake_up.remote() 替代 collective_rpc
        # 确保信号通过 DP coordinator 广播到所有 EngineCore 进程
        if self.config.free_cache_engine and self._ensure_server_handle():
            await self.server_handle.wake_up.remote(tags=tags)

    async def release(self):
        """Release weights and kv cache in GPU memory."""
        # 同样直接调用 server_handle.sleep.remote()
        if self.config.free_cache_engine and self._ensure_server_handle():
            await self.server_handle.sleep.remote()
```

verl/workers/rollout/vllm_rollout/vllm_async_server.py

vLLM 服务端，修复了 _sleep_hybrid 和 wake_up，替换 collective_rpc 为 engine.sleep/wake_up，支持 HYBRID 模式下的 wake_up，清理了 dead code。

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

    async def wake_up(self, tags: list[str] | None = None):
        if self.node_rank != 0:
            return
        if self.rollout_mode == RolloutMode.HYBRID:
            # 之前这里抛出 ValueError，现在直接调用 engine.wake_up()
            # engine.wake_up() 通过 DP coordinator 广播到所有 EngineCore 进程（所有 DP shard）
            # 不同于 collective_rpc 只到达单个 shard 内的 TP worker
            await self.engine.wake_up(tags=tags or self._get_wake_up_tags())
            await self.engine.reset_prefix_cache()
        elif self.rollout_mode == RolloutMode.COLOCATED:
            await self.engine.wake_up(tags=self._get_wake_up_tags())
            await self.engine.reset_prefix_cache()
        elif self.rollout_mode == RolloutMode.STANDALONE:
            logger.info("skip wake_up in standalone mode")

    async def _sleep_hybrid(self):
        """HYBRID sleep: 使用 engine.sleep() 替代 engine.collective_rpc("sleep")

        原因: collective_rpc 仅到达单个 DP shard 内的 TP workers,
        导致其他 DP shard 的模型权重未释放，引起 FSDP 训练 OOM。
        engine.sleep() 通过 DPAsyncLLM 协调器广播到所有 EngineCore 进程。
```

```
    """
    # 根据 LoRA 或 NPU 环境决定 sleep level
    if 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) # 关键替换
    if _VLLM_VERSION >= version.parse("0.17.0"):
        await self.engine.reset_encoder_cache()
```

评论区精华

争议点：是否应彻底消除所有 collective_rpc 调用

- reviewer wuxibin89指出：ServerAdapter 中 resume() 和 release() 仍然通过 collective_rpc 调用 sleep/wake_up，建议彻底消除 collective_rpc。
- author dafu-wu先回应：ServerAdapter.release() 是死代码（SPMD 模式遗留），实际活跃路径已通过 _sleep_hybrid 修复。随后接受建议，在最新提交中将 ServerAdapter 的 resume 和 release 直接改为调用 server_handle.wake_up.remote 和 server_handle.sleep.remote，彻底消除了 collective_rpc 绕路。

未解决疑虑：是否有其他仍然使用 collective_rpc 的路径

- 经审查，目前的变更已覆盖所有活跃的 sleep/wake_up 调用路径；但 _execute_method 仍保留给其他方法（如 update_weights_from_ipc）使用，不存在 DP 传播风险。

reviewer gemini-code-assist[bot] 的高优先级评论

- 指出 _sleep_hybrid 修复不完整，因为 ServerAdapter 层仍使用 collective_rpc。author 确认并修复。
- 是否应彻底消除所有 collective_rpc 调用 (design): 在 ServerAdapter.resume() 和 release() 中直接调用 server_handle.wake_up.remote() 和 server_handle.sleep.remote()，消除了 collective_rpc 绕路。
- collective_rpc 中 hasattr 守卫是否为死代码 (correctness): 已移除，后续提交清理。
- ServerAdapter.release() 是否是死代码 (question): 确认为死代码，后续也做了清理（调用 server_handle.sleep.remote()）。

风险与影响

- 风险：### 低风险
- 核心路径变更但单行等效替换: engine.collective_rpc("sleep", kwargs=...) 替换为 engine.sleep(level=...)，语义上都是调用 engine 的 sleep，区别仅在于传播范围。现有 DP=1 场景行为不变。
- ServerAdapter 层函数调用变更: _execute_method → 直接 server_handle.xxx.remote()，但 _execute_method 对于 sleep/wake_up 只有一个 caller，且新路径直接绕过 collective_rpc 的远程调用开销，性能一致。
- 没有新增依赖或配置: 纯内部重构，不改变用户 API 或配置项。

- 回归风险: 若未来有其他代码依赖 ServerAdapter.release()（当前死代码），可能因突然生效而暴露问题；但作者和 reviewer 均确认其已废弃。
- 未覆盖测试: 缺少 DP>1 的多 GPU 测试，但手动验证通过。
- 影响：### 影响范围
- 用户场景: 修复 HYBRID 模式 + data_parallel_size > 1 时的 CUDA OOM 崩溃；此场景之前无法运行，修复后可正常工作。
- 系统影响: 消除 collective_rpc 在 sleep/wake_up 上的不必要间接层，降低远程调用开销（虽然是微秒级）。
- 团队影响: 为后续 DP 相关功能提供了正确模式；清理了死代码，降低维护成本。
- 影响程度: 中。修复了一个阻塞性 bug，但仅影响特定配置（HYBRID 模式 + DP>1），默认 DP=1 用户无感知。
- 风险标记：核心路径变更，缺少多 GPU 测试覆盖，死代码清理可能暴露遗留问题

关联脉络

- PR #5716 [rollout] refactor: flowgrpo 相关重构：PR body 指出原始代码在 flowgrpo 重构前使用的是 engine.sleep(level=2)，重构时被错误改为 collective_rpc，导致 DP>1 时 OOM。此 PR 回退到正确的 engine.sleep 路径。
- PR #3902 Memory usage increased after sleeping: PR body 提到的类似问题，但发生在 Ascend NPU + 旧 SPMD 架构上，场景不同但相关。