PR #6716 完整报告

verl-project/verl

[trainer] fix: use FullyAsyncLLMServerClient for async trainer

合并时间：2026-06-13 01:10

原文链接：http://prhub.com.cn/verl-project/verl/pull/6716

执行摘要

- 一句话：迁移 FullyAsyncLLMServerClient 至核心模块，供异步 trainer 使用
- 推荐动作：建议阅读此 PR 以了解异步训练中 rollout 客户端如何统一，并关注未解决的 LSP 问题。设计上迁移到核心模块是合理的演进方向。

功能与动机

Follow up https://github.com/verl-project/verl/pull/6710, use FullyAsyncLLMServerClient for async trainer colocate_async/separate_async to automatically resume on abort.

实现拆解

1. 类迁移：在 verl/workers/rollout/llm_server.py 中新增 FullyAsyncLLMServerClient 类，继承 LLMServerClient，覆盖 generate 方法实现带中断恢复的生成逻辑。同时从 verl/experimental/fully_async_policy/fully_async_rollouter.py 中删除该类的原始定义，减少代码冗余。
2. 导入更新：在 fully_async_rollouter.py 中将导入从 LLMServerClient 改为 FullyAsyncLLMServerClient，并移除不再需要的工具函数引用（如 normalize_token_ids、rollout_trace_op 等）。
3. 训练器集成：
   - trainer_colocate_async.py：新增 get_llm_client 方法，通过 self.llm_server_manager.get_client(client_cls=FullyAsyncLLMServerClient) 获取客户端。
   - trainer_separate_async.py：修改已有 get_llm_client 方法，传入 client_cls=FullyAsyncLLMServerClient 参数，替代无参调用。
4. 测试适配：更新 tests/checkpoint_engine/test_special_server_adapter.py 中的导入路径，从 verl.experimental... 改为 verl.workers.rollout.llm_server，确保测试能正确引用。

关键文件：

- verl/experimental/fully_async_policy/fully_async_rollouter.py（模块 实验模块；类别 source；类型 core-logic；符号 FullyAsyncLLMServerClient, generate）：移除了 FullyAsyncLLMServerClient 类的原始定义和导入，避免了代码重复，简化了实验模块。

- verl/workers/rollout/llm_server.py（模块 rollout 层；类别 source；类型 core-logic；符号 FullyAsyncLLMServerClient, generate）：新增了 FullyAsyncLLMServerClient 类作为核心模块，是所有异步 trainer 的 rollout 客户端。
- verl/trainer/ppo/v1/trainer_colocate_async.py（模块 异步训练器；类别 source；类型 core-logic；符号 get_llm_client）：新增 get_llm_client 方法，显式使用 FullyAsyncLLMServerClient，是异步训练器的关键集成点。
- verl/trainer/ppo/v1/trainer_separate_async.py（模块 异步训练器；类别 source；类型 dependency-wiring；符号 get_llm_client）：修改 get_llm_client 方法以传入 client_cls 参数，对齐异步逻辑。
- tests/checkpoint_engine/test_special_server_adapter.py（模块 测试；类别 test；类型 test-coverage）：更新导入路径以匹配新的类位置，确保测试通过。

关键符号：FullyAsyncLLMServerClient.generate, PPOTrainerColocateAsync.get_llm_client, PPOTrainerSeparateAsync.get_llm_client

关键源码片段

verl/trainer/ppo/v1/trainer_colocate_async.py

新增 get_llm_client 方法，显式使用 FullyAsyncLLMServerClient，是异步训练器的关键集成点。

```
# verl/trainer/ppo/v1/trainer_colocate_async.py
from verl.workers.rollout.llm_server import FullyAsyncLLMServerClient

@register_trainer("colocate_async")
class PPOTrainerColocateAsync(PPOTrainer):
    # ... other methods ...

    def get_llm_client(self):
        """Get the LLM server client for rollout generation."""
        # 返回完全异步客户端，支持 partial rollout 自动恢复
        return self.llm_server_manager.get_client(client_cls=FullyAsyncLLMServerClient)
```

评论区精华

review 中 gemini-code-assist[bot] 提出两个关于 Liskov 替换原则的问题：

- 子类 FullyAsyncLLMServerClient.generate 方法签名未包含 **kwargs，可能导致运行时 TypeError。
- 未在 super().generate() 调用中转发 **kwargs，建议添加。 当前 PR 中未采纳这两条建议，LSP 风险未解决。
- 子类 generate 方法签名缺少 **kwargs (design): 未在代码中采纳，LSP 风险存在。
- 未转发 **kwargs 到 super().generate (design): 未采纳。

风险与影响

- 风险：主要风险是子类 *generate* 方法未接收并转发 **kwargs，违反 LSP，若未来父类 LLMServerClient.*generate* 新增可选参数或被多态调用时，可能导致运行时错误。此外，类移动到新模块后，所有引用点均已更新，但需确认无遗漏的间接引用。
- 影响：影响范围局限于异步训练模式（*colocate_async* 和 *separate_async*），不影响同步训练或其他模块。开发者可以更统一地管理 rollout 客户端逻辑，降低维护成本。用户无直接感知。
- 风险标记：违反 Liskov 替换原则，未转发 kwargs 可能导致运行时错误

关联脉络

- PR #6710 [trainer] feat: add unify trainer abstraction for sync and async training: 本 PR 是 #6710 的后续，使用统一训练器抽象中的 FullyAsyncLLMServerClient。