PR #2604 完整报告

radixark/miles

fix(rollout): raise server readiness timeout to 120s

合并时间：2026-08-25 04:24

原文链接：http://prhub.com.cn/radixark/miles/pull/2604

执行摘要

- 一句话：将 router/session-server 就绪超时提升至 120s，修复 CI 偶发失败
- 推荐动作：该 PR 值得快速合并，修复了 CI 偶发失败且改动简单、注释清晰。对于读者，它展示了如何通过根因分析定位偶发故障并采用合理的超时策略。无需深读，但可以留意后续是否将常量提取为可配置参数。

功能与动机

在 CI 运行 32090638792 的 stage-c-2-gpu-h200 job 中，test_glm5_744b_a40b_4layer_r3.py 因 'Server at 172.18.0.2:4077 not ready after 30s' 失败（由 start_router 的 wait_for_server_ready(timeout=30) 引起）。该 job 中 6/7 个测试通过，失败属于偶发。根因是 spawn 子进程需要重新导入重型依赖链（典型 13 秒），而 CI 瞬时卡顿可能使启动超过 30 秒预算。PR body 详细排除了端口冲突（日志中无 EADDRINUSE，且端口 4077 在其他测试中被成功复用），确认问题为启动时间超时。

实现拆解

本 PR 的修改集中在 miles/ray/rollout/router_manager.py，主要分两步：

1. 新增模块常量：在文件顶部引入 _SERVER_READY_TIMEOUT_SECS = 120，并注释说明原因（spawn 上下文重新导入重型 transformers/megatron 链，CI 瞬时卡顿可能超过 30 秒预算）。
2. 替换硬编码超时：将 start_router 和 start_session_server 中的 wait_for_server_ready(..., timeout=30) 改为使用新常量 _SERVER_READY_TIMEOUT_SECS。

此改动不影响快速失败：wait_for_server_ready 内部的 process.is_alive() 检查会在子进程死亡时立即抛出异常，因此宽限预算只影响仍存活但启动缓慢的情形。

测试与兼容性：未新增专门测试，但作者运行了 tests/fast/ray/rollout/test_router_manager.py（11 个测试通过）。该改动改变了超时行为，但不涉及数据结构或外部接口，向后兼容。

关键文件：

- miles/ray/rollout/router_manager.py（模块 路由管理；类别 source；类型 entrypoint；符号 _SERVER_READY_TIMEOUT_SECS, start_router, start_session_server）：核心修改文件：将 router/session-server 就绪超时从硬编码 30s 提升为模块常量 120s，修复

CI 偶发启动超时导致的训练失败。

关键符号：start_router, start_session_server

关键源码片段

miles/ray/rollout/router_manager.py

核心修改文件：将 router/session-server 就绪超时从硬编码 30s 提升为模块常量 120s，修复 CI 偶发启动超时导致的训练失败。

```
import copy
import logging
import multiprocessing
import random
import uuid

from sglang_router.launch_router import RouterArgs

from miles.rollout.session.server import run_session_server
from miles.router.router import run_router as run_miles_router
from miles.utils.http_utils import _wrap_ipv6, find_available_port, get_host_info, is_port_
available
from miles.utils.http_utils import run_router as run_sglang_router
from miles.utils.http_utils import wait_for_server_ready

logger = logging.getLogger(__name__)

# 就绪预算：spawn 子进程需要重新导入重型 transformers/megatron 链（CI 中约 13 秒），
# 瞬时卡顿可能超过 30 秒，故提升到 120 秒。
_SERVER_READY_TIMEOUT_SECS = 120


def start_router(args, *, has_pd_disaggregation: bool = False, force_new: bool = False) ->
tuple[str, int]:
    """启动 sgl router 或 miles router，返回 (router_ip, router_port)。"""
    if not force_new and args.sglang_router_ip is not None:
        return args.sglang_router_ip, args.sglang_router_port

    router_ip = _wrap_ipv6(get_host_info()[1])
    if force_new:
        router_port = find_available_port(random.randint(3000, 4000))
    else:
        router_port = args.sglang_router_port
        if router_port is None:
            router_port = find_available_port(random.randint(3000, 4000))

    if args.use_miles_router:
        assert not has_pd_disaggregation, "miles router does not support PD disaggregation."
        run_router = run_miles_router
```

```
        router_args = copy.copy(args)
        router_args.sglang_router_ip = router_ip
        router_args.sglang_router_port = router_port
    else:
        run_router = run_sglang_router
        router_args = RouterArgs.from_cli_args(args, use_router_prefix=True)
        router_args.host = router_ip
        router_args.port = router_port
        router_args.prometheus_port = find_available_port(random.randint(4000, 5000))
        router_args.log_level = "warn"
        router_args.request_timeout_secs = args.sglang_router_request_timeout_secs
        if args.sglang_router_policy:
            router_args.policy = args.sglang_router_policy
        if has_pd_disaggregation:
            router_args.pd_disaggregation = True
        logger.info(f"Launch router with args: {router_args}")

    port = router_port
    if not is_port_available(port):
        raise RuntimeError(
            f"Port {port} is already in use — a stale router process may still be running. "
            f"Run 'pkill -9 python' to kill it, then retry."
        )

    # 使用 spawn（而非 fork）：避免子进程继承 Ray actor 的线程 / 终结器（如 wandb
    服务线程）导致死锁。
    process = multiprocessing.get_context("spawn").Process(
        target=run_router,
        args=(router_args,),
    )
    process.daemon = True
    process.start()
    wait_for_server_ready(router_ip, router_port, process, timeout=_SERVER_READY_TIMEOUT_
    SECS)
    logger.info(f"Router launched at {router_ip}:{router_port}")
    return router_ip, router_port
```

评论区精华

PR 的 code review 评论为 0，仅有两条 APPROVED（来自 Shi-Dong 和 yueming-yuan）以及一条 claude[bot] 的自动提示（告知仓库配置了手动评审）。因此没有实质性的技术讨论。不过 PR body 中包含细致的根因分析和逻辑推理，如端口冲突排除、进程存活检查的作用等，可作为讨论精华。

- 暂无高价值评论线程

风险与影响

- 风险：主要风险是放宽超时可能使故障检测延迟，即当子进程卡死但未退出时，训练任务会等待最多 120 秒而非 30 秒。不过由于子进程由 multiprocessing 启动且设置了 daemon，若子进程僵死，父进程仍可能通过 is_alive() 检查。另外，该超时值对所有调用统一生效，如果未来子进程启动模式变化（如增加更多重型导入），可能需要重新评估。整体而言，风险较低。
- 影响：影响范围仅涉及 rollout 模块的路由器与会话服务器启动逻辑。对用户而言，CI 中偶发的启动超时失败减少，提升训练任务的稳定性。对系统而言，可能增加启动等待时间（最坏情况 120 秒），但只在异常场景下发生。对团队而言，这是一个轻量级修复，无接口变更，维护成本低。
- 风险标记：超时延长可能导致故障检测延迟，缺少直接针对超时常量的单元测试

关联脉络

- PR #1313 RDT weight sync: GPU->GPU zero copy transfer through SGLang Ray actor backend: 同属 rollout 模块，且修改了 miles/ray/rollout 下的相关文件，可能影响 router 启动逻辑。
- PR #2729 fix(ci): use local UCX transports for RDT test: 同为 CI 稳定性修复，且涉及 rollout 相关测试，与本 PR 目标一致。