PR #7613 完整报告

verl-project/verl

[rollout] fix: randomize least-loaded tie-break so sessions do not avalanche onto one replica

合并时间：2026-08-31 14:41

原文链接：http://prhub.com.cn/verl-project/verl/pull/7613

执行摘要

- 一句话：负载均衡 tie-break 改随机选择，修复 session 雪崩到单 replica
- 推荐动作：值得精读。这是一个「一行修复背后有完整系统思维」的范本：PR body 对雪崩机制的解释（idle 池 + 确定性 tie-break + sticky 缓存放大）清晰展示了多因素交互导致的问题，值得学习的是它如何界定修复边界——只随机化首轮选择、保留 sticky 与 full_determinism 两条既有保证。同时，Codex 评论揭示了测试与实现细节耦合的教训，读者可借机检查自身测试是否过度绑定内部选择顺序。建议：合并后优先确认声明的 CPU 测试文件去向，并跟进 agent_loop 既有测试的修复，避免 CI 长期 flaky。

功能与动机

PR body 明确指出问题根因：池空闲时所有 replica 的 inflight 计数相同，candidates[0] 每次都解析到字典序第一个 replica，而 sticky-session 缓存会把该请求的所有轮次都钉在同一 replica 上，导致「new sessions avalanche onto a single replica while the rest of the pool sits idle」。作者还检索了同类 PR（load balancer tie）确认没有先例，说明这是一个未被注意的负载均衡正确性缺陷。

实现拆解

整个变更集中在一个文件、两处小改动，配合 PR body 中声称的测试方案，可按以下步骤拆解：

1. 新增标准库导入：在 verl/workers/rollout/llm_server.py 的 import 区加入 import random，紧跟在 import os 之后，保持标准库导入顺序（asyncio、logging、os、random、typing、uuid）。
2. 改造 tie-break 逻辑：在 GlobalRequestLoadBalancer.acquire_server 的普通选择分支（full_determinism=False 时）中，将 server_id = candidates[0] 改为 server_id = random.choice(candidates)，并补充注释说明随机化与 sticky-session 缓存的交互：只有 session 第一轮受随机选择影响，后续轮次仍走 self._request_id_to_server 缓存，因此前缀缓存（prefix-cache）局部性得以保留。
3. 保持确定性路径隔离：full_determinism=True 分支（按 hash(request_id) % len(self._servers) 路由）完全未动，需要跨运行可复现路由的调用方不受影响；release_server、add_servers、remove_servers 等其余方法也未改动。

4. 测试配套（存在疑点）：PR body 声称新增了 tests/workers/rollout/test_llm_server_load_balance_on_cpu.py，包含 4 个用例：fan-out（4 replica、200 次 acquire/release 循环，main 上会失败）、sticky sessions preserved、least-loaded still wins、full_determinism unchanged，并给出 4 passed 结果。但实际合并 diff 仅包含 llm_server.py 一个文件，测试文件并未进入本 PR，这一点需要在风险中明确标注。
5. 协作与收尾：两个 commit 中，第一个是作者 YZH0216 的完整修复，第二个是维护者 wuxibin89 提交的仅含 "fix" 消息的提交，具体内容材料未提供，无法确认是否包含测试相关调整。

关键文件：

- verl/workers/rollout/llm_server.py（模块 负载均衡；类别 source；类型 core-logic；符号 GlobalRequestLoadBalancer.acquire_server）：本 PR 唯一变更文件，包含全部核心改动：GlobalRequestLoadBalancer.acquire_server 的 tie-break 从固定 candidates[0] 改为 random.choice(candidates)，并新增 import random。该负载均衡器被所有 AgentLoopWorker 共享，是全局 session 路由的决策点。

关键符号：GlobalRequestLoadBalancer.acquire_server

关键源码片段

verl/workers/rollout/llm_server.py

本 PR 唯一变更文件，包含全部核心改动：GlobalRequestLoadBalancer.acquire_server 的 tie-break 从固定 candidates[0] 改为 random.choice(candidates)，并新增 import random。该负载均衡器被所有 AgentLoopWorker 共享，是全局 session 路由的决策点。

```
# 摘自 verl/workers/rollout/llm_server.py
# GlobalRequestLoadBalancer.acquire_server 完整实现（head 版本）：
# 该负载均衡器被所有 AgentLoopWorker 共享，负责为新请求挑选托管 replica，并维护 sticky 会话粘性。

def acquire_server(self, request_id: str) -> tuple[str, ray.actor.ActorHandle]:
    """为请求获取一个 server（sticky session + 最小负载选择），原子返回 (server_id, actor_handle)。"""
    # 1. 优先走 sticky session：同一请求（多轮对话）的所有轮次都复用第一次选中的 replica，
    # 这是前缀缓存局部性（prefix-cache hit）的关键来源。
    if request_id in self._request_id_to_server:
        server_id = self._request_id_to_server[request_id]
        # 若该 replica 仍在活跃池中，直接复用并增加 in-flight 计数
        if server_id in self._inflight_requests:
            self._inflight_requests[server_id] += 1
            return server_id, self._servers[server_id]
        # replica 已被移除，清掉过期缓存条目并重新选择
        del self._request_id_to_server[request_id]

    # 2. 池为空时无法服务任何请求
    if not self._inflight_requests:
        raise RuntimeError("No available servers in load balancer")
```

```
        # 3. full_determinism 模式：按 request_id 哈希路由，保证跨运行可复现；
        # 最小负载选择依赖异步到达时序（运行间不稳定），因此被完全绕过。
        if self._full_determinism:
            server_id = list(self._servers)[hash(request_id) % len(self._servers)]
        else:
            # 4. 普通模式：在 " 同等最闲 " 的 replica 中均匀随机选择，而不是固定取 candidates[0]。
            # 池空闲时（启动时必然、低并发时常见）所有 inflight 计数都为 0，
            # 固定取第一个会把每个新 session 通过 sticky 缓存钉在同一 replica 上，
            # 其余 replica 长期闲置（雪崩效应）。
            # 随机 tie-break 只影响 session 的第一轮，后续轮次仍走 sticky 缓存，
            # 因此前缀缓存局部性不受影响。
            min_count = min(self._inflight_requests.values())
            candidates = [sid for sid, count in self._inflight_requests.items() if count == min_count]
            server_id = random.choice(candidates)

        # 5. 记录选择结果到 sticky 缓存，并增加该 replica 的 in-flight 计数
        self._request_id_to_server[request_id] = server_id
        self._inflight_requests[server_id] += 1
        return server_id, self._servers[server_id]
```

评论区精华

本次 review 只有一条实质性评论，来自 Codex 自动审查机器人，但价值很高：

- Codex 在 verl/workers/rollout/llm_server.py:112 处标记 P1 问题：随机化会让 tests/experimental/agent_loop 目录下既有测试依赖运气——test_new_requests_route_to_least_loaded 期望最终选中 s2（约 1/3 概率），test_removed_server_invalidates_sticky_session 和 test_get_inflight_count 假设初始选中 s0（约 1/2 概率）。若不捕获选定 ID 或显式控制随机选择，这些测试在多数运行中会失败。
- 该评论没有收到作者或维护者的公开回复，但从合并结果看，PR 最终被 wuxibin89 approve 并合入，第二个 commit "fix" 是否处理了这些测试脆弱性无法从材料中确认。
- 这是「测试断言绑定实现细节」的典型反例：测试不应假设 tie-break 的确定性顺序，除非该顺序是 API 契约的一部分。
- 随机 tie-break 使既有 agent_loop 测试依赖运气 (testing): PR 已由 wuxibin89 approve 并合入，但未见对既有测试的公开修改跟进；第二个 commit "fix" 是否包含相关调整无法从材料确认，测试脆弱性问题在合并时仍悬而未决。

风险与影响

- 风险：
  1. 既有测试回归风险（最具体）：Codex 点名 tests/experimental/agent_loop 下的 test_new_requests_route_to_least_loaded、test_removed_server_invalidates_sticky_session、test_get_inflight_count 三个测试分别依赖旧 tie-break 的确定结果（s2、s0），随机化后失败概率约 1/3 与 1/2。这些测试在 vLLM 工作流中运行，合入后 CI 大概率出现 flaky。

2. 声明测试未落地：PR body 明确描述并验证了 test_llm_server_load_balance_on_cpu.py（4 passed），但实际 diff 不包含该文件。fan-out 回归用例的缺失意味着「雪崩」问题没有自动化守护，未来可能被无意改回。
3. 行为变更影响面：random.choice 使用全局随机状态，非 full_determinism 模式下路由不再可复现。对负载均衡目标是改进，但任何依赖「同等最闲时固定选第一个」的调用方（例如监控脚本或隐式假设）会观察到行为变化。
4. 并发安全性：GlobalRequestLoadBalancer 既作为普通 Python 类也可被 ray.remote 包装为 actor 串行执行，random.choice 无状态竞争问题，此风险可排除。
5. 前缀缓存影响可控：仅 session 首轮路由被随机化，后续轮次仍走 sticky 缓存，prefix-cache locality 不受破坏，这一点 PR body 与代码注释都做了说明，风险低。

- 影响：
  - 系统 / 用户影响：多 replica rollout 部署（尤其 AgentLoopWorker 多并发 session 场景）下，新 session 的初始分布从「全部涌向同一 replica」变为「在同等最闲 replica 间均匀分散」，显著改善低并发与启动阶段的负载均衡，提升整体吞吐与 replica 利用率；对单 replica 部署无任何影响。
  - 团队影响：需要同步加固 tests/experimental/agent_loop 下依赖旧 tie-break 顺序的测试，否则 CI 会间歇性变红；后续涉及负载均衡的改动需明确「确定性路由 = full_determinism」的边界，避免再次引入隐性确定性假设。
  - 影响程度评估：代码改动量极小（+2/-1），但处于所有 AgentLoopWorker 共享的全局调度路径，属于核心路径行为变更，值得合并后持续观察 agent_loop 相关 CI 稳定性。
  - 风险标记：既有测试依赖旧 tie-break 行为，PR body 所述测试未进入 diff，随机路由与确定性测试冲突，核心调度路径行为变更

关联脉络

- PR #7565 [rollout] fix: surface vLLM prefix-cache hit counts in TokenOutput: 改动同一文件 verl/workers/rollout/llm_server.py，涉及 TokenOutput 与 rollout 统计逻辑，与本 PR 同属 rollout 调度与统计基础设施的演进。
- PR #7518 [rollout, ci] fix: make agent-loop tests fully deterministic: 该 PR 致力于消除 agent-loop 测试的不确定性，与本 PR 引入随机 tie-break 方向相反，恰好构成对照：确定性路由被收敛到 full_determinism 开关下，其余路径允许随机化，值得放在一起理解项目的确定性取舍边界。
- PR #7632 [vllm] fix: raise max_num_batched_tokens to max_model_len when chunked prefill is disabled: 该 PR 修改了 tests/experimental/agent_loop/test_basic_agent_loop.py，正是 Codex 评论点名的脆弱测试所在目录与文件，本 PR 的随机化会放大这些测试的不稳定性。