PR #50965 完整报告

vllm-project/vllm

[Bugfix] Fix get_open_port() livelock on DP-reserved ports and cover get_open_ports_list

合并时间：2026-08-07 22:39

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/50965

执行摘要

- 一句话：修复 DP 端口保留段内 VLLM_PORT 死循环并补齐批量端口过滤
- 推荐动作：值得精读。第一，根因分析教科书式清晰：确定性输入 + 无状态重试 = 死循环，修复利用「升序扫描从区间末尾开始不可能再落入区间」的数学性质，用一次有界重扫替换整个循环，是低成本高收益的小规模设计决策。第二，测试中用守护线程 + 超时断言把「死循环回归」转化为「快速失败」，该模式可复用到其他 hang 类缺陷的回归测试。第三，多个并行 PR 解决同一问题时，通过 commit 保留原作者署名、在正文致谢未纳入方案的做法，是社区协作的良好示范。

功能与动机

issue #50024 明确指出：设置 VLLM_PORT 为数据并行保留窗口内的值时，vllm serve --data-parallel-size N 会在启动时永久挂起且无任何日志或报错。根因是 get_open_port() 的 while True 循环在 VLLM_PORT 固定时每次得到同一端口，拒绝后重试同一端口，永不推进。issue 还指出 #36191 已为 get_open_ports_list 引入 start_port/max_attempts 机制，但 get_open_port 的调用点从未接上，修复机器就在同一文件内。

实现拆解

修复按以下步骤落地：

1. 新增共享 helper _get_reserved_port_range()：将「数据并行主进程保留端口区间」的计算从 get_open_port() 中提取出来（vllm/utils/network_utils.py）。未设置 VLLM_DP_MASTER_PORT 时返回空 range(0)，调用方无需处理 None，两个调用点共享同一语义。
2. 重写 get_open_port() 控制流：去掉 while True 重试循环，改为「先取一个候选端口 → 判断是否落入保留区间 → 落入则从 reserved_port_range.stop 起调用 _get_open_port(start_port=..., max_attempts=1000) 重扫一次」。利用升序扫描从区间末尾开始不可能再次落入该区间的性质，一次有界重扫即可替代整个循环，同时覆盖确定性路径（VLLM_PORT 固定）和临时端口路径。
3. 为 get_open_ports_list() 补齐保留区间过滤：该函数的 VLLM_PORT 分支原本完全没有保留窗口过滤，可能把 DP master 要绑定的端口发出去，造成确定性的未来冲突。现在每轮扫描后同样做区间判断，命中则从区间末尾跳档重扫，并保持 next_port 推进逻辑不变。

4. 测试配套：tests/utils_/test_network_utils.py 新增 _call_with_timeout() 守护线程辅助函数，并加入 5 个回归测试：VLLM_PORT == VLLM_DP_MASTER_PORT（原死循环场景）、VLLM_PORT 位于区间内部、get_open_ports_list(5) 全部避开保留区间、临时端口落入区间时被重扫替换、未配置保留区间时行为不变。守护线程 + 超时断言保证若死循环回归，测试会快速失败而不是挂死整个套件。

关键文件：

- vllm/utils/network_utils.py（模块 网络工具；类别 source；类型 core-logic；符号 _get_reserved_port_range, get_open_port, get_open_ports_list）：核心修复文件：新增 _get_reserved_port_range() 共享 helper，重写 get_open_port() 的 while True 重试循环为一次越界重扫，并为 get_open_ports_list() 的 VLLM_PORT 分支补齐此前缺失的 DP 保留区间过滤。
- tests/utils_/test_network_utils.py（模块 回归测试；类别 test；类型 test-coverage；符号 _call_with_timeout, target, test_get_open_port_vllm_port_in_dp_reserved_range, test_get_open_port_skips_reserved_dp_master_ports）：回归测试文件：新增 _call_with_timeout 守护线程辅助与 5 个测试用例，覆盖 VLLM_PORT 在保留区间起点 / 内部、批量列表过滤、临时端口落入区间、无保留区间配置等场景，防止死循环回归挂死测试套件。

关键符号：_get_reserved_port_range, get_open_port, get_open_ports_list, _call_with_timeout

关键源码片段

vllm/utils/network_utils.py

核心修复文件：新增 _get_reserved_port_range() 共享 helper，重写 get_open_port() 的 while True 重试循环为一次越界重扫，并为 get_open_ports_list() 的 VLLM_PORT 分支补齐此前缺失的 DP 保留区间过滤。

```
# vllm/utils/network_utils.py 核心修复片段


def _get_reserved_port_range() -> range:
    """计算数据并行主进程保留的端口区间（未设置时返回空 range）。

    当前为 DP master 预留 10 个端口（[VLLM_DP_MASTER_PORT, +10)），
    实际使用 2 个；返回空 range 可以让调用方免去 None 判断。
    """
    if "VLLM_DP_MASTER_PORT" not in os.environ:
        return range(0)
    dp_master_port = envs.VLLM_DP_MASTER_PORT
    return range(dp_master_port, dp_master_port + 10)


def get_open_port() -> int:
    """获取一个可绑定的端口，且避开 DP master 的保留窗口。
```

```
    旧实现是 while True 重试：候选端口若落入保留区间会被拒绝，
    但 VLLM_PORT 固定时 _get_open_port() 每次返回同一个端口，于是死循环。
    新实现只做一次判断：从区间末尾（stop）重新扫描必然不会再落入区间，
    因此一次有界重扫即可替代整个重试循环。
    """
    reserved_port_range = _get_reserved_port_range()
    port = _get_open_port()
    if port in reserved_port_range:
        port = _get_open_port(start_port=reserved_port_range.stop, max_attempts=1000)
    return port


def get_open_ports_list(count: int = 5) -> list[int]:
    """批量获取唯一端口；VLLM_PORT 分支同样过滤 DP 保留窗口。"""
    ports_set = set[int]()
    if envs.VLLM_PORT is not None:
        reserved_port_range = _get_reserved_port_range()
        next_port = envs.VLLM_PORT
        for _ in range(count):
            port = _get_open_port(start_port=next_port, max_attempts=1000)
            if port in reserved_port_range:
                # 与 get_open_port 相同策略：从保留区间末尾继续扫描
                port = _get_open_port(
                    start_port=reserved_port_range.stop, max_attempts=1000
                )
            ports_set.add(port)
            next_port = port + 1
        return list(ports_set)
    else:
        while len(ports_set) < count:
            ports_set.add(get_open_port())
    return list(ports_set)
```

评论区精华

该 PR 几乎没有 review 技术交锋：claude[bot] 注明 fork 拉取请求自动 review 被禁用，需维护者手动触发；ZJY0516 直接 APPROVED，未留下技术质疑。真正的技术讨论集中在 PR body 的 "Relationship to existing PRs"：作者明确披露 #50049（首个引入越过区间扫描思路与回归测试）、#50043（引入 max_attempts=1000 有界扫描）两个并行 PR 的贡献，以 commit 形式保留原作者署名；#50025 采用相同重扫思路但未纳入，作者在正文致谢。最终 commit 将控制流简化为「共享 helper + 一次越界重扫」，去掉循环与新增异常路径，并补齐 get_open_ports_list 过滤与测试。

- fork 自动审查被禁用，改为维护者直接审核 (other): ZJY0516 未触发 bot review，直接 APPROVED。
- 与 #50049/#50043/#50025 并行方案的合并与署名 (design): 合并时保留两份原作者 credit，最终方案统一为「共享 helper + 一次越界重扫」。

风险与影响

- 风险：行为变更集中在端口分配路径，具体风险点：
  - 未设置 VLLM_DP_MASTER_PORT 的用户：_get_reserved_port_range() 返回空 range(0)，任何端口都不在区间内，逻辑与原实现等价，回归风险低。
  - 端口选择结果可能变化：设置保留区间后，若候选端口命中区间，现在会跳档到区间末尾之后；这是修复意图，但依赖该端口分配行为的调用方（如显式端口绑定的部署脚本）可能观察到端口变化。
  - 边界行为：VLLM_PORT 恰好等于 reserved_range.stop 时不在右开区间内，行为正常；若从 stop 起连续 1000 个端口均不可用，会按既有 _get_open_port 逻辑抛 RuntimeError，相比旧实现的无限循环是更可诊断的失败。
  - TOCTOU 竞态：bind 与返回之间端口仍可能被其他进程抢占，多进程并发启动时该问题存在，但非本次变更引入，测试也未覆盖并发场景。
  - 影响：用户侧：修复了 vllm serve --data-parallel-size N 在 VLLM_PORT 与 VLLM_DP_MASTER_PORT 冲突时静默挂死的启动缺陷——该问题在 0.26.0 镜像中可复现（挂起超过 120 秒无输出），端到端验证显示修复后 DP server 在一分钟内恢复健康。系统侧：改动仅限 vllm/utils/network_utils.py 的两个函数与一个共享 helper，不涉及模型执行、调度、注意力等核心路径，影响面窄；get_open_ports_list() 的端口分布可能变化，但方向是消除与 DP master 的确定性冲突。团队侧：已附带完整回归测试和端到端验证记录，维护成本低，后续合入风险可控。
  - 风险标记：核心启动路径变更，端口分配语义变化，回归测试覆盖充分

关联脉络

- PR #50049 [Bugfix] get_open_port(): resume scan past DP reserved range: PR 正文说明本 PR 第一个 commit 来自该 PR（作者 hclsys，署名保留），它首次引入越过保留区间扫描的思路和回归测试，但保留重试循环且未覆盖 get_open_ports_list。
- PR #50043 [Bugfix] get_open_port(): bound scan with max_attempts=1000: 本 PR 第二个 commit 来源（作者 ferkans-amir 署名保留），引入 max_attempts=1000 有界扫描；其逐端口推进方式被最终方案的区间跳跃取代，max_attempts 保留在最终代码中。
- PR #50025 [Bugfix] get_open_port() VLLM_PORT path rescan: 同一问题的第三个并行 PR，采用相同重扫思路但未纳入本 PR，未覆盖 get_open_ports_list，作者在正文中致谢。