PR #42585 完整报告

vllm-project/vllm

[Bugfix][V1] Fix TOCTOU race causing intermittent `EADDRINUSE` on multi-API-server DP startup

合并时间：2026-05-27 05:06

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/42585

执行摘要

- 一句话：修复多 API 服务器启动时的端口竞态条件
- 推荐动作：值得精读，尤其适合需要了解多进程端口安全分配和 ZMQ 启动流程的开发者；设计决策（延迟端口绑定 vs 预分配）及 review 中的讨论具有通用借鉴意义。

功能与动机

父进程通过 get_open_port() (bind + close) 预分配临时端口，但子进程直到 5–25 秒后才真正绑定 ZMQ socket。在此窗口内，其他进程（包括 vLLM 自身其他端口分配）可能占用该端口，导致 EADDRINUSE。PR body 明确指出："Fix: Apply the same bind-in-child + report-back-via-Pipe pattern that DPCoordinator already uses."

实现拆解

1. 修改 get_engine_client_zmq_addr（vllm/v1/utils.py）：移除内部的 get_open_port() 调用，直接返回 tcp://host:port（port 为 0 或指定值）；将端口选择的职责交给调用者，调用者绑定后通过 getsockopt(zmq.LAST_ENDPOINT) 获取实际端口。
2. 在 APIServerProcessManager 中引入 Pipe 机制（vllm/v1/utils.py）：每个 API 服务器子进程启动时，在 client_config 中附带一个 multiprocessing.Pipe 的写端 (actual_address_pipe)。子进程绑定 ZMQ socket 后，通过该 Pipe 发送实际端点字典；父进程通过新增的 gather_actual_addresses() 方法收集所有子进程的地址，并在超时或子进程崩溃时抛出明确异常。
3. 在 MPClient.__init__ 中添加双向适配（vllm/v1/engine/core_client.py）：如果 client_config 中包含 actual_address_pipe，则在绑定 socket 后通过 getsockopt 获取实际端点并发送；如果不存在 Pipe（例如单进程模式），则在绑定后自行解析 tcp://host:0 占位符，就地更新 addresses 对象后再启动引擎。
4. 修改 get_engine_zmq_addresses（vllm/v1/engine/utils.py）：新增 defer_api_server_ports 参数（默认 True），生成 tcp://host:0 占位符；同时为 handshake_address 保留旧的 port=0 自动分配行为。
5. 集成到 CLI 启动入口（vllm/entrypoints/cli/serve.py）：在创建 APIServerProcessManager 后立即调用 gather_actual_addresses()，将实际地址写回 addresses.inputs 和 addresses.outputs，再进入 launch_core_engines 上下文管理器的引擎握手阶段。Rust 前端通过 defer_api_server_ports=False 选择禁用此模式。

关键文件：

- vllm/v1/utils.py（模块 进程管理；类别 source；类型 core-logic；符号 get_engine_client_zmq_addr, gather_actual_addresses, APIServerProcessManager.init, APIServerProcessManager.shutdown）：核心变更：移除 get_engine_client_zmq_addr 中的端口预分配，新增 gather_actual_addresses 方法和 Pipe 初始化逻辑。
- tests/entrypoints/test_api_server_process_manager.py（模块 测试；类别 test；类型 test-coverage；符号 defer_addresses_stub_worker, test_gather_actual_addresses_end_to_end, test_gather_actual_addresses_child_crash_before_report）：新增端到端测试，覆盖正常路径和子进程崩溃场景，验证 Pipe 收集地址的正确性。
- vllm/v1/engine/utils.py（模块 引擎工具；类别 source；类型 core-logic；符号 get_engine_zmq_addresses, _addr）：修改 get_engine_zmq_addresses 以支持延迟端口分配参数，生成 tcp://host:0 占位符。
- vllm/v1/engine/core_client.py（模块 客户端核心；类别 source；类型 core-logic；符号 MPClient.init, AsyncMPClient.init）：在 MPClient 中实现 Pipe 报告和占位符解析，确保两种模式都正确获取实际端点。
- vllm/v1/engine/coordinator.py（模块 协调器；类别 source；类型 cleanup；符号 DPCoordinator.init）：简化 bind_address 函数，移除冗余的 get_open_port 调用，使用统一的 get_engine_client_zmq_addr。
- vllm/entrypoints/cli/serve.py（模块 CLI 入口；类别 source；类型 core-logic）：集成 gather_actual_addresses 调用，将实际地址反馈给引擎握手流程。
- vllm/v1/engine/async_llm.py（模块 异步引擎；类别 source；类型 cleanup）：类型注解调整以配合 client_addresses 的泛化。

关键符号：get_engine_client_zmq_addr, gather_actual_addresses, APIServerProcessManager.init, APIServerProcessManager.shutdown, get_engine_zmq_addresses, MPClient.init, AsyncMPClient.init, DPCoordinator.init, run_multi_api_server

关键源码片段

vllm/v1/utils.py

核心变更：移除 get_engine_client_zmq_addr 中的端口预分配，新增 gather_actual_addresses 方法和 Pipe 初始化逻辑。

```
# vllm/v1/utils.py (gather_actual_addresses)
def gather_actual_addresses(
    self,
    timeout: float = 30.0,
) -> tuple[list[str], list[str]]:
    """从每个子进程的 pipe 中读取实际绑定的端点。

    等待所有子进程报告它们的 ROUTER/PULL 地址。如果子进程在
```

```
    报告之前退出，pipe 会 EOF，此时抛出 ``RuntimeError``。
    """
    if self._address_pipes is None:
        raise RuntimeError("cannot gather addresses without deferring port assignment")

    start = time.monotonic()
    actual_inputs: list[str | None] = [None] * len(self._address_pipes)
    actual_outputs: list[str | None] = [None] * len(self._address_pipes)

    # 使用 ``ConnectionPoll`` 来在超时内轮询所有 pipe。
    pending = set(range(len(self._address_pipes)))
    pipes = self._address_pipes[:] # 复制，因为我们在循环中会关闭它们

    while pending and (time.monotonic() - start < timeout):
        for i in list(pending):
            if pipes[i].poll(0):
                msg = pipes[i].recv()
                actual_inputs[i] = msg["input_address"]
                actual_outputs[i] = msg["output_address"]
                pipes[i].close()
                pending.remove(i)
        if pending:
            time.sleep(0.01)

    # 关闭所有剩余的 pipe
    for i in pending:
        pipes[i].close()

    if pending:
        failed_indices = sorted(pending)
        raise RuntimeError(
            f"{len(pending)} API server(s) did not report addresses: "
            f"indices {failed_indices}"
        )

    return (
        [addr for addr in actual_inputs if addr is not None], # type: ignore[misc]
        [addr for addr in actual_outputs if addr is not None], # type: ignore[misc]
    )
```

tests/entrypoints/test_api_server_process_manager.py

新增端到端测试，覆盖正常路径和子进程崩溃场景，验证 Pipe 收集地址的正确性。

```
# tests/entrypoints/test_api_server_process_manager.py
# 模块级工作进程，必须可被 multiprocessing.spawn 导入（不能是闭包或嵌套函数）
def defer_addresses_stub_worker(listen_address, sock, args, client_config):
    """绑定 ROUTER/PULL socket 并报告实际端点。"""
    ctx = zmq.Context()
    try:
```

```
        # 以 ``tcp://host:0`` 方式绑定，内核自动分配空闲端口
        in_sock = make_zmq_socket(
            ctx, client_config["input_address"], zmq.ROUTER, bind=True
        )
        out_sock = make_zmq_socket(
            ctx, client_config["output_address"], zmq.PULL, bind=True
        )
        # 通过 ``getsockopt(zmq.LAST_ENDPOINT)`` 获取实际端点
        pipe = client_config["actual_address_pipe"]
        try:
            pipe.send({
                "input_address": in_sock.getsockopt(zmq.LAST_ENDPOINT).decode(),
                "output_address": out_sock.getsockopt(zmq.LAST_ENDPOINT).decode(),
            })
        finally:
            pipe.close()
    finally:
        in_sock.close(linger=0)
        out_sock.close(linger=0)
        ctx.term()

# 由 pytest 调用的端到端测试
def test_gather_actual_addresses_end_to_end():
    host = "127.0.0.1"
    num_servers = 4
    # 生成占位符地址 tcp://host:0
    placeholder_inputs = [
        get_engine_client_zmq_addr(local_only=False, host=host)
        for _ in range(num_servers)
    ]
    placeholder_outputs = [
        get_engine_client_zmq_addr(local_only=False, host=host)
        for _ in range(num_servers)
    ]
    for addr in placeholder_inputs + placeholder_outputs:
        assert addr == f"tcp://{host}:0", addr

    sock = socket.socket()
    manager = APIServerProcessManager(
        listen_address=f"tcp://{host}:0",
        sock=sock,
        args="test_args",
        num_servers=num_servers,
        input_addresses=placeholder_inputs,
        output_addresses=placeholder_outputs,
        target_server_fn=defer_addresses_stub_worker,
    )
    try:
        actual_inputs, actual_outputs = manager.gather_actual_addresses(timeout=15.0)
```

```
    finally:
        manager.shutdown()
        sock.close()

    assert len(actual_inputs) == num_servers
    assert len(actual_outputs) == num_servers
    for addr in actual_inputs + actual_outputs:
        # 验证地址是完整格式且端口不是 0
        scheme, parsed_host, port = split_zmq_path(addr)
        assert scheme == "tcp"
        assert port != 0
```

评论区精华

统一新方式 vs 保留旧方式的争论：Robert 认为应全量改用新方式，移除旧分支，使代码更清晰。njhill 解释 APIServerProcessManager 目前仅用于多 API 服务器场景，而 AsyncLLM 直接使用引擎时不需要，因此保留抽象是合理的。两人同意后续可单独 PR 优化。

类型注解争议：gemini-code-assist[bot] 指出 client_addresses 从 dict[str, str] 变为包含 Connection 对象，导致 # type: ignore 的压制。作者最终将 client_addresses 类型改为 dict[str, Any] 以恢复类型安全。

关键 Bug 发现：gemini 发现 get_engine_client_zmq_addr 的行为变更破坏了 handshake_address 的逻辑（引擎 handshake 仍需要预先分配端口）。vadiklyutiy 在后续提交中修复，保留 port=0 自动分配路径。

socket 绑定误报：gemini 警告 output_socket 未显式绑定就调用 getsockopt，作者标记为“False positive”——因为 make_zmq_socket 在 bind=True 时已绑定，此处 socket 正是以 bind=True 创建的。

资源泄漏：gemini 建议在关闭 Pipe 后立即释放父端句柄，并在 shutdown 中统一清理 _address_pipes。作者已采纳。

- 统一新方式 vs 保留旧方式 (design): 当前设计可接受，后续可单独 PR 移除 context manager。
- client_addresses 类型注解问题 (style): 作者将类型改为 dict[str, Any] 并移除 type: ignore。
- get_engine_client_zmq_addr 行为变更破坏 handshake_address (correctness): 作者在后续提交中为 handshake_address 保留 port=0 自动分配行为（通过 get_open_port() 填充）。
- Socket 未绑定就调用 getsockopt 的担忧 (correctness): 作者标记为“False positive”——make_zmq_socket 在 bind=True 时已绑定 socket，此处 socket 正是用 bind=True 创建。
- Pipe 资源泄漏 (performance): 作者在 gather_actual_addresses 中已关闭，并在 shutdown 中添加清理。

风险与影响

- 风险：回归风险（低）：对于 IPC 路径（local_only=True）无影响；对于 Rust 前端（已设置 defer_api_server_ports=False）行为不变。get_engine_client_zmq_addr 的行为变更已适配所有内部调用者。

兼容性风险（中）：get_engine_client_zmq_addr 不再自动分配端口，如果外部代码直接使用此函数并依赖其端口分配，可能失效。但该函数被标记为内部 API。

资源泄漏（已解决）：gather_actual_addresses 和 shutdown 方法都包含了 Pipe 关闭逻辑，通过 CI 验证。

时序风险（低）：gather_actual_addresses 设置了 15 秒超时，子进程提前崩溃会立即检测到并抛出 RuntimeError，避免死锁。测试覆盖了子进程崩溃场景。

- 影响：用户：多 API 服务器 DP 启动的 EADDRINUSE 错误彻底消失；在极端端口压力测试中（仅剩 100 个端口），修复后通过率从 1/28 提升至 50/50。

系统：端口分配模式更健壮，消除了因端口占用导致的启动失败。

团队：代码复杂度略有增加，但模式统一（与 DPCoordinator 一致），后续维护成本较低。

- 风险标记：核心路径变更，端口 0 占位符依赖，Rust 前端未适配

关联脉络

- PR #40596：Issue 评论中提到 #40596 做了相同的修复，但未合并。
- PR #41983：Issue 评论中提到 #41983 也做了相同的修复，但未合并。