

PR #53437 完整报告

vllm-project/vllm

[CI][AMD] Preserve diagnostics for unwritable checkouts

合并时间: 2026-09-02 16:18

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/53437>

执行摘要

- 一句话: 修复 AMD CI 在 checkout 不可写时丢失诊断信息的问题。
- 推荐动作: 此 PR 值得 CI 维护者和基础设施团队精读。虽然它不涉及核心推理逻辑, 但其设计模式——为关键操作添加多级 fallback、处理环境兼容性、确保最终信息可见性——对于构建健壮的 CI/CD 流程很有价值。实现细节 (如安全地上传文件、清理资源) 也值得借鉴。

功能与动机

根据 PR body 描述, 一个失败的 AMD CI 作业在清理阶段无法创建 `artifacts/amd-gpu-diagnostics/.../diagnostics.log` 文件, 原因是 checkout 目录不可写, 导致后续配置的 artifact scan 找不到任何文件。此变更的核心动机是确保即使在 checkout 权限受限的环境中, 关键的诊断信息也不会丢失, 从而提升 CI 的可诊断性和问题排查效率。

实现拆解

1. 理解现有流程与定位问题点: 分析现有的 `collect_rocm_failure_diagnostics` 函数, 识别其在创建 `diagnostics_path` 文件时, 若因权限问题失败, 整个诊断流程会停止并输出警告, 导致关键信息 (如 ROCm 版本、GPU 状态、内存事件等) 丢失。
2. 引入 fallback 存储机制: 在 `collect_rocm_failure_diagnostics` 函数中, 当检测到无法在 checkout 目录创建诊断文件时, 尝试在系统临时目录中创建一个同结构路径的文件 (通过 `mktemp -d`)。将此状态记录在新变量 `diagnostics_fallback_root` 和 `diagnostics_storage` 中。
3. 实现独立的上传函数 `upload_failure_diagnostics_artifact`: 新增此函数, 负责从指定根目录上传诊断文件。该函数会智能查找 `buildkite-agent` 二进制文件 (考虑 `BUILDKITE_BIN_PATH`、系统路径和固定 `/workspace` 路径), 并通过 `timeout` 命令限制上传操作, 设置调试追踪为关闭状态。关键兼容性处理包括: 通过检查 `artifact upload --help` 输出来判断是否支持 `--literal` 和 `--delimiter` 参数, 仅对路径中含有通配符或分隔符字符的文件使用这些选项, 以支持旧版 Buildkite agent。
4. 集成上传与错误处理: 在诊断收集的主体流程末尾, 检查 `diagnostics_fallback_root` 是否存在。若存在, 则调用 `upload_failure_diagnostics_artifact` 尝试上传。上传成功, 则更新汇总信息; 若失败或超时, 则将诊断内容直接输出到作业日志 (`cat "${diagnostics_path}"`) 作为最终保障, 并更新汇总信息为“仅作业日志”。无论上传结果如何, 都尝试清理临时目录。

5. 增强状态记录与输出：在诊断汇总输出中，新增 diagnostics_storage 字段，明确标识诊断信息是存储在 checkout 还是临时回退位置。同时，将实际的 artifact 路径或上传状态记录在汇总表中。

关键文件：

- .buildkite/scripts/hardware_ci/run-amd-test.sh（模块 CI 脚本；类别 infra；类型 infrastructure；符号 upload_failure_diagnostics_artifact, collect_rocm_failure_diagnostics）：此脚本是 AMD CI 诊断收集的核心入口，本次所有变更均在此文件中完成，实现了 fallback 存储、独立上传和增强的状态报告。

关键符号：upload_failure_diagnostics_artifact, collect_rocm_failure_diagnostics

关键源码片段

.buildkite/scripts/hardware_ci/run-amd-test.sh

此脚本是 AMD CI 诊断收集的核心入口，本次所有变更均在此文件中完成，实现了 fallback 存储、独立上传和增强的状态报告。

```
# 从源码片段重新整理的新上传函数，体现了对旧版 agent 的兼容性处理和安全上传逻辑
upload_failure_diagnostics_artifact() {
    local upload_root=$1 upload_path=$2 agent_bin=""
    local agent_upload_help=""
    local -a upload_options=()

    # 智能查找 buildkite-agent 二进制文件，兼容不同环境配置
    if [[ -n "${BUILDKITE_BIN_PATH:-}" ]] \
        && -x "${BUILDKITE_BIN_PATH}/buildkite-agent" ]; then
        agent_bin="${BUILDKITE_BIN_PATH}/buildkite-agent"
    else
        agent_bin=$(command -v buildkite-agent 2>/dev/null || true)
    fi
    if [[ -z "${agent_bin}" ]] && -x /workspace/buildkite-agent ]; then
        agent_bin=/workspace/buildkite-agent
    fi
    # 基础检查：agent 和 Job ID 必须存在，timeout 命令可用
    [[ -n "${agent_bin}" ]] && -n "${BUILDKITE_JOB_ID:-}" ]] || return 1
    command -v timeout >/dev/null 2>&1 || return 1

    # 兼容性处理：对于老版本 AMD runner，若路径包含通配符等特殊字符，
    # 需要检查 agent 是否支持 --literal 和 --delimiter 选项
    if [[ ! "${upload_path}" =~ ^[A-Za-z0-9_][A-Za-z0-9_-]*(/[A-Za-z0-9_][A-Za-z0-9_-]*)*$ ]]; then
        agent_upload_help="${agent_bin} artifact upload --help 2>&1 || true)
        if [[ "${agent_upload_help}" != *"--literal"* \
            || "${agent_upload_help}" != *"--delimiter"* ]]; then
            return 1
        fi
        upload_options=(--literal --delimiter "")
    fi
```

```
# 执行上传：关闭调试追踪，设置超时，确保操作在受限环境下也能尝试完成
(
  cd "${upload_root}" || exit 1
  BUILDKITE_AGENT_DEBUG=false \
  BUILDKITE_AGENT_DEBUG_HTTP=false \
  BUILDKITE_AGENT_TRACE_HTTP=false \
  BUILDKITE_AGENT_LOG_LEVEL=error \
  timeout --kill-after=2s "${amd_diagnostics_upload_timeout_seconds}s" \
  "${agent_bin}" artifact upload \
  "${upload_options[@]}" "${upload_path}"
) >/dev/null 2>&1
}
```

评论区精华

此 PR 的 review 讨论较少，主要由 [claude\[bot\]](#) 机器人评论触发，[tjtanaa](#) 进行了批准。讨论的核心价值体现在 commit 历史的迭代中：

1. 兼容性与健壮性设计：作者在后续提交（如 [a1b3306a](#) 和 [68cc85b7](#)）中，专门增强了函数以处理旧版 Buildkite agent 的兼容性问题，通过检查 agent 的 `--help` 输出动态决定上传参数，体现了对部署环境差异的深入考虑。
2. 优雅降级与最终保障：方案不仅提供了临时存储和直接上传的 fallback，还确保在所有自动上传手段失败后，将诊断信息直接打印到作业日志，这是一种重要的“最终保障”设计，确保信息在任何情况下都能被看到。

结论：PR 的设计重点在于提高 CI 诊断流程的健壮性，通过多级 fallback 机制和对不同 agent 版本的适配，确保诊断信息在各种异常条件下（如权限问题、agent 版本老旧）仍可获取。

- 对旧版 Buildkite agent 的兼容性处理 (design): 通过动态检查 agent 能力来设置上传选项，实现了对新旧 agent 的广泛兼容，这是一种健壮的设计决策。

风险与影响

- 风险：
 1. CI 流程变更风险：修改了 AMD CI 的核心诊断收集脚本 `run-amd-test.sh`，若存在逻辑疏漏，可能影响失败作业的诊断信息收集。但变更集中在故障清理路径，正常成功路径未受影响。
 2. 临时文件清理风险：新增了 `mktemp -d` 创建临时目录的逻辑。虽然最后有 `rm -rf` 清理，但如果上传或后续操作耗时过长或脚本异常中断，可能残留临时文件。不过使用了 `mktemp -d` 和独立的 `rm` 命令，风险可控。
 3. 兼容性验证风险：对旧版 Buildkite agent 的兼容性（通过检查 `--help` 输出）是在当前测试环境中验证的。未来若出现行为不同的新旧 agent 混合，可能存在未预料到的边界情况。
 4. 超时设置：上传操作设置了 `amd_diagnostics_upload_timeout_seconds`（默认 20 秒）的超时。若网络或 agent 极慢，可能导致上传超时，回退到日志输出，这是预期的降级

行为。

- 影响：

1. 直接影响：主要影响 AMD CI (Buildkite) 作业，特别是那些因环境权限问题导致 checkout 目录不可写的失败作业。这些作业现在能可靠地将诊断信息作为 artifact 上传，或至少输出到作业日志。
2. 用户（开发者）影响：对 vLLM 开发者，尤其是维护 ROCm 后端的团队，排查 AMD 相关 CI 失败将更加容易，因为关键的诊断信息（GPU 状态、驱动信息、内存事件等）更容易获取。
3. 系统影响：提高了 CI 基础设施的可靠性和可维护性。该变更不影响任何 vLLM 运行时性能、模型功能或 API。
4. 团队影响：为其他平台的 CI 脚本提供了健壮的错误处理和 fallback 模式的设计参考。
 - 风险标记：CI 脚本核心路径变更，临时文件清理可靠性

关联脉络

- PR #54895 [CI] Use PR head label for Buildkite branch to avoid main collision: 同属 CI/Buildkite 基础设施优化系列，解决不同的 CI 流程可靠性问题（分支命名冲突 vs. 诊断信息丢失）。
- PR #54753 [CI] Shard basic model initialization tests: 同属 CI 效率与可靠性改进，但侧重点不同（测试分片优化 vs. 失败诊断保全）。