PR #52822 完整报告

vllm-project/vllm

[ROCm][CI] Add AMD CI Pull-Request Commands

合并时间：2026-08-19 06:16

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/52822

执行摘要

- 一句话：新增 AMD 专属 PR 评论命令，独立映射 Buildkite amd-ci 流水线
- 推荐动作：值得精读，尤其是 run_ci_command.py 中通过 frozenset 分集合 + 映射函数统一控制流的做法，以及为不同 CI 行为差异设计的 blocks_new_run、is_comment_triggered_build 等函数，对理解多流水线 CI 命令系统的扩展模式有参考价值。建议关注后续是否有将触发条件从手工命令列表改为读取脚本常量的重构机会，以消除重复维护。

功能与动机

作者在 Issue 评论中明确说明动机："The motivation for this is that sometimes we want to launch a full nightly but only for AMD, like for an AITER version bump or something. And we do not want to impose any extra burden on upstream CI pipeline." 即 AMD 团队在升级 AITER 等依赖时需要单独触发 AMD nightly，而不希望占用上游 CI 资源。PR body 进一步指出需要显式区分命令与流水线映射：新命令精确指向 Buildkite amd-ci 流水线，同时保留原有 /ci 命令对上游 ci 流水线的语义，并要求命令白名单与流水线选择解耦，避免任意评论前缀选中流水线。

实现拆解

1. 扩展命令定义与集合划分：在 .github/workflows/scripts/run_ci_command.py 中新增 5 个 AMD 命令常量（COMMAND_RUN_AMD_CI、COMMAND_RUN_AMD_CI_ALL、COMMAND_RUN_AMD_CI_NIGHTLY、COMMAND_RETRY_AMD_FAILED、COMMAND_CANCEL_AMD_CI），并建立 UPSTREAM_CI_COMMANDS、AMD_CI_COMMANDS、RETRY_COMMANDS、CANCEL_COMMANDS、ALL_CI_COMMANDS 等 frozenset，parse_command 改为基于 ALL_CI_COMMANDS 精确匹配，拒绝带后缀或大小写变体（如 /amd-ci run please、/AMD-CI run）。
2. 引入流水线映射函数：新增 pipeline_for_command、ci_name_for_command、run_command_for_command、retry_command_for_command，将命令映射为对应 Buildkite 流水线名称（amd-ci 或 ci）、人读名称（AMD CI 或 CI）及对应的 run/retry 提示命令。handle_run_ci、handle_retry_failed 等入口改用这些映射函数生成提示与回复，使上游与 AMD 路径共用一套控制流。
3. 适配 AMD CI 行为差异：

- 新增 is_comment_triggered_build（检测 meta_data.github-comment-id）和 blocks_new_run；AMD 命令下，仅当已存在活跃的命令触发构建（state 为 active 且非 blocked）时才阻止新建，而普通 webhook 触发的 blocked 构建不阻塞显式评论触发；
- /amd-ci retry 只针对当前 PR head 的构建重试，若无当前 head 构建则直接提示用户先运行 /amd-ci run，避免使用不稳定步骤 key 做跨 commit 选择重试；
- /amd-ci cancel 同时检查 PR 分支与 owner-qualified fork 分支的构建。

4. 工作流与文档配套：.github/workflows/run-ci-command.yml 的触发条件扩为包含 5 个 AMD 命令，并新增 BUILDKITE_AMD_PIPELINE: amd-ci 环境变量；new_pr_bot.yml 与 docs/contributing/README.md 同步更新命令指引，说明 /ci run 对应 upstream CI、/amd-ci run 对应 AMD CI only。
5. 单元测试补强：在 test_run_ci_command.py 中新增 8 个测试用例，覆盖流水线选择（test_commands_select_only_their_configured_pipeline）、AMD blocked 构建不阻塞（test_amd_run_ignores_blocked_builds）、评论触发去重（test_amd_run_deduplicates_comment_triggered_active_build）、环境变量传递（test_amd_run_variants_set_buildkite_environment）、未授权作者提示（test_unapproved_author_gets_amd_specific_guidance）、retry 仅针对当前 head 及缺失构建提示（test_amd_ci_retry_retries_only_the_current_head_build、test_amd_ci_retry_requires_a_build_for_the_current_head）、cancel 分支与 fork 分支处理（test_amd_ci_cancel_handles_command_and_fork_webhook_branches）。同时 FakeBuildkite 增加 list_requests 记录以断言分支 /commit/metadata 查询参数。

关键文件：

- .github/workflows/scripts/run_ci_command.py（模块 CI 命令；类别 infra；类型 infrastructure；符号 pipeline_for_command, ci_name_for_command, run_command_for_command, retry_command_for_command）：核心实现文件。新增 AMD 命令常量、命令集合划分、流水线映射函数（pipeline_for_command / ci_name_for_command 等），并重构 handle_run_ci / handle_retry_failed / handle_cancel 等入口以支持 AMD 专属行为。
- .github/workflows/scripts/test_run_ci_command.py（模块 CI 测试；类别 test；类型 test-coverage；符号 test_commands_select_only_their_configured_pipeline, test_amd_run_ignores_blocked_builds, test_amd_run_deduplicates_comment_triggered_active_build, test_amd_run_variants_set_buildkite_environment）：测试配套文件，新增 8 个针对 AMD 命令的测试用例，覆盖流水线选择、blocked 构建忽略、评论触发去重、环境变量、未授权提示、retry 语义与 cancel 分支处理，是验证实现行为的关键依据。
- .github/workflows/run-ci-command.yml（模块 工作流；类别 infra；类型 infrastructure）：Workflow 入口变更：在 job 触发条件中显式加入 5 个 /amd-ci 命令，并新增 BUILDKITE_AMD_PIPELINE 环境变量，让脚本能读到 amd-ci 流水线名。
- .github/workflows/new_pr_bot.yml（模块 工作流；类别 infra；类型 infrastructure）：新 PR 自动评论文案同步更新，明确区分 upstream CI 与 AMD CI 的命令用法，减少贡献者误用。
- docs/contributing/README.md（模块 贡献文档；类别 docs；类型 documentation）：贡献文档同步补充 /amd-ci 命令说明，与 new_pr_bot.yml 一起构成面向贡献者的完整命令使

用指南。

关键符号：pipeline_for_command, ci_name_for_command, run_command_for_command, retry_command_for_command, is_comment_triggered_build, blocks_new_run, parse_command, handle_run_ci, handle_retry_failed, notify_authorized, create_retry_build_payload

关键源码片段

.github/workflows/scripts/run_ci_command.py

核心实现文件。新增 AMD 命令常量、命令集合划分、流水线映射函数（pipeline_for_command / ci_name_for_command 等），并重构 handle_run_ci / handle_retry_failed / handle_cancel 等入口以支持 AMD 专属行为。

```
# .github/workflows/scripts/run_ci_command.py
# 命令常量：与上游 /ci 系列并列新增 /amd-ci 系列
COMMAND_RUN_AMD_CI = "/amd-ci run"
COMMAND_RUN_AMD_CI_ALL = "/amd-ci run all"
COMMAND_RUN_AMD_CI_NIGHTLY = "/amd-ci run nightly"
COMMAND_RETRY_AMD_FAILED = "/amd-ci retry"
COMMAND_CANCEL_AMD_CI = "/amd-ci cancel"

# 将命令划分为两个互斥集合，分别对应上游 ci 流水线与 AMD amd-ci 流水线
UPSTREAM_CI_COMMANDS = frozenset({
    COMMAND_RUN_CI, COMMAND_RUN_CI_ALL, COMMAND_RUN_CI_NIGHTLY,
    COMMAND_RETRY_FAILED, COMMAND_CANCEL_CI,
})
AMD_CI_COMMANDS = frozenset({
    COMMAND_RUN_AMD_CI, COMMAND_RUN_AMD_CI_ALL, COMMAND_RUN_AMD_CI_
    NIGHTLY,
    COMMAND_RETRY_AMD_FAILED, COMMAND_CANCEL_AMD_CI,
})
RETRY_COMMANDS = frozenset({COMMAND_RETRY_FAILED, COMMAND_RETRY_AMD_FAILED}
)
CANCEL_COMMANDS = frozenset({COMMAND_CANCEL_CI, COMMAND_CANCEL_AMD_CI})
ALL_CI_COMMANDS = UPSTREAM_CI_COMMANDS | AMD_CI_COMMANDS

# 只有完整精确匹配的命令才会被接受，杜绝任意前缀选中流水线
def parse_command(body: str) -> str | None:
    if body in ALL_CI_COMMANDS:
        return body
    return None

# 核心映射：命令 -> Buildkite 流水线名，用于建构建时选择 amd-ci 或 ci
def pipeline_for_command(
    command: str,
    *,
    amd_ci_pipeline: str = "amd-ci",
    upstream_ci_pipeline: str = "ci",
```

```
) -> str:
    if command in AMD_CI_COMMANDS:
        return amd_ci_pipeline
    if command in UPSTREAM_CI_COMMANDS:
        return upstream_ci_pipeline
    raise ValueError(f"Unsupported CI command: {command}")

# 判定某构建是否由评论触发：AMD 去重只认评论触发的活跃构建
def is_comment_triggered_build(build: Mapping[str, Any]) -> bool:
    metadata = build.get("meta_data") or {}
    return bool(metadata.get("github-comment-id"))

# AMD 与上游的“是否阻止新构建”策略不同：
# 上游只要存在活跃构建就阻止；AMD 则忽略 webhook 阻塞构建，
# 仅当存在评论触发的活跃构建（非 blocked）时才阻止，从而允许显式评论覆盖阻塞状态。
def blocks_new_run(command: str, build: Mapping[str, Any]) -> bool:
    if command in AMD_CI_COMMANDS:
        return (
            is_comment_triggered_build(build)
            and build.get("state") in ACTIVE_BUILD_STATES
            and build.get("state") != "blocked"
        )
    return is_active_build(build)
```

.github/workflows/scripts/test_run_ci_command.py

测试配套文件，新增 8 个针对 AMD 命令的测试用例，覆盖流水线选择、blocked 构建忽略、评论触发去重、环境变量、未授权提示、retry 语义与 cancel 分支处理，是验证实现行为的关键依据。

```
# .github/workflows/scripts/test_run_ci_command.py
# 验证命令到流水线的映射是严格白名单：每个命令只能落到 ci 或 amd-ci
def test_commands_select_only_their_configured_pipeline(self) -> None:
    cases = (
        (COMMAND_RUN_CI, "ci"),
        (COMMAND_RUN_CI_ALL, "ci"),
        (COMMAND_RUN_CI_NIGHTLY, "ci"),
        (COMMAND_RETRY_FAILED, "ci"),
        (COMMAND_CANCEL_CI, "ci"),
        (COMMAND_RUN_AMD_CI, "amd-ci"),
        (COMMAND_RUN_AMD_CI_ALL, "amd-ci"),
        (COMMAND_RUN_AMD_CI_NIGHTLY, "amd-ci"),
        (COMMAND_RETRY_AMD_FAILED, "amd-ci"),
        (COMMAND_CANCEL_AMD_CI, "amd-ci"),
    )
    for command, expected_pipeline in cases:
        with self.subTest(command=command):
            self.assertEqual(pipeline_for_command(command), expected_pipeline)
    # 任意前缀不能选中流水线
    with self.assertRaisesRegex(ValueError, "Unsupported CI command"):
```

```
        pipeline_for_command("/amd-ci run arbitrary-pipeline")

    # AMD 场景：webhook 触发的 blocked 构建不阻塞显式评论触发的新 build
    # 无论 blocked 构建是否带 github-comment-id，都应允许新建
    def test_amd_run_ignores_blocked_builds(self) -> None:
        for metadata in ({}, {"github-comment-id": "98"}):
            with self.subTest(metadata=metadata):
                github = FakeGitHub()
                blocked_build = {
                    "blocked": True,
                    "created_at": "2026-08-18T01:00:00Z",
                    "meta_data": metadata,
                    "number": 122,
                    "pull_request": {"id": 42},
                    "state": "blocked",
                    "web_url": "https://buildkite.example/amd-ci/builds/122",
                }
                buildkite = FakeBuildkite([[], [blocked_build]])

                run(make_event(COMMAND_RUN_AMD_CI_ALL), github, buildkite)

                self.assertEqual(len(buildkite.created_builds), 1)
                self.assertEqual(buildkite.created_builds[0]["env"]["RUN_ALL"], "1")

    # 已有评论触发的活跃 AMD 构建时，应去重并提示用户
    def test_amd_run_deduplicates_comment_triggered_active_build(self) -> None:
        github = FakeGitHub()
        command_build = {
            "blocked": False,
            "created_at": "2026-08-18T01:00:00Z",
            "meta_data": {"github-comment-id": "98"},
            "number": 122,
            "pull_request": {"id": 42},
            "source": "api",
            "state": "running",
            "web_url": "https://buildkite.example/amd-ci/builds/122",
        }
        buildkite = FakeBuildkite([[], [command_build]])

        run(make_event(COMMAND_RUN_AMD_CI_ALL), github, buildkite)

        self.assertEqual(buildkite.created_builds, [])
        self.assertIn("AMD CI is already running", github.comments[0])
```

评论区精华

review 主要来自机器人自动评论与维护者批准：claude[bot] 仅提示手动 review 机制，无实质技术讨论；khluu 直接 APPROVED。作者 AndreasKaratzas 在 Issue 评论中解释了动机（AMD 需要独立 nightly），并在 PR 内以 /ci run 实际触发了上游 CI（Buildkite CI #84462）。

由于本 PR 没有人工 review 评论线程，关键技术权衡（如 AMD 重试为何不做跨 commit 选择性重试、blocked 构建为何不阻塞）主要由 PR body 的说明和测试用例体现，未见到被质疑或推翻的讨论。

- AMD 独立 CI 命令的动机与资源隔离 (question): 新增 /amd-ci 系列命令，精确映射到 amd-ci 流水线，与上游 /ci 命令隔离。
- AMD retry 不支持跨 commit 选择性重试 (design): 接受该限制，并在测试中锁定该行为（test_amd_ci_retry_requires_a_build_for_the_current_head）。
- AMD webhook blocked 构建不阻塞评论触发 (design): 通过 is_comment_triggered_build 与 blocks_new_run 实现差异逻辑，测试覆盖 blocked 元数据有无 comment-id 两种情况。

风险与影响

- 风险：
  1. 命令冲突风险：新增 /amd-ci 前缀命令与现有 /ci 命令并存，若用户记错前缀（如 /amd- ci、/amd-ci retry 带尾随空格）会被 parse_command 拒绝，属于预期内行为；但若未来新增命令未同步加入 ALL_CI_COMMANDS，会导致白名单遗漏而静默失效。
  2. CI 资源与去重逻辑：blocks_new_run 对 AMD 命令仅拦截评论触发的活跃构建，不拦截 webhook 触发的 blocked 构建。若 webhook 构建实际仍在消耗资源（例如管道处于 queued 但构建状态不是 blocked），可能造成重复运行；不过测试已覆盖 blocked 场景，整体风险可控。
  3. 重试语义差异：AMD retry 强制仅针对当前 head，且无当前 head 构建时要求先 /amd-ci run。若用户期望重试旧 commit 的失败任务，该行为会被拒绝，可能造成一定使用摩擦，但 PR body 已说明这是因 AMD 测试步骤缺乏稳定 key 的刻意权衡。
  4. Workflow 触发条件与脚本硬编码：run-ci-command.yml 的触发条件手工列出每个命令，与脚本内 ALL_CI_COMMANDS 集合重复维护；未来新增命令若只改脚本不改 workflow，会导致评论触发不到 workflow。当前改动已同步更新，但维护上存在隐性耦合。
  5. 文档一致性：new_pr_bot.yml 与 docs/contributing/README.md 的命令说明依赖人工同步，本次已同步更新，但后续若命令集合再变化需注意更新两处文档。 - 影响：影响范围集中在 CI/CD 与开发者体验层面，不涉及模型推理、内核或运行时逻辑。对 AMD 团队而言，可直接通过 /amd-ci run nightly 触发 AMD 专属 nightly，无需占用上游 CI 资源，显著提升 AITER 等依赖升级的迭代效率；对普通贡献者而言，命令提示文案更清晰地区分了 upstream CI 与 AMD CI，减少了误用。对系统而言，多引入一条 Buildkite 流水线映射，但通过精确命令白名单与去重逻辑控制资源消耗。测试覆盖较完整，改动集中在配置与脚本层，对既有 /ci 命令的行为有少量文案调整但没有语义破坏。 - 风险标记：命令白名单与 workflow 触发条件重复维护，AMD retry 跨 commit 能力受限，AMD blocked 构建不拦截可能重复触发，文档与命令集合同步风险

关联脉络

- PR #52659 [CI] Standardize test job labels by device: 同为 CI 基础设施调整，涉及 Buildkite 作业标签与流水线配置，与本 PR 的 CI 命令体系同属 CI 运维演进方向。

- PR #52810 [CI][ROCm] Prevent Git maintenance races during shallow fetches: 同为 ROCm CI 稳定性改进，与本 PR 的 AMD CI 命令目标一致，都是优化 AMD/ROCm 侧的 CI 体验。
- PR #52593 [Build] Propagate vLLM version to Rust binaries: 涉及 CI/ 构建基础设施与版本注入，与本 PR 一样属于 build/CI 工具链的配套演进，可能共享 CI 配置上下文。