PR #50328 完整报告

vllm-project/vllm

[CI/Build][AMD] Install triton_kernels via CMake

合并时间：2026-07-31 11:05

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/50328

执行摘要

- 一句话：按平台分派构建 triton_kernels，修复 gfx950 GPT-OSS 崩溃
- 推荐动作：值得粗读（约 5 分钟）：核心价值在于两点——CMake 按目标设备分派同一依赖的不同源码来源 + 供应链固定到 commit 的实践，以及运行时去掉 vendored 回退、统一外部包导入的契约简化。需要特别关注的风险点是 except ImportError 静默返回可能掩盖依赖缺失，以及非 ROCm 平台去掉 vendored 回退后的兼容性；若团队维护 ROCm 多架构，建议跟进 fork commit 的更新节奏与 gfx90a/gfx942 回归验证。

功能与动机

PR body 明确说明：'Starting with ROCm 7.14 we will install dependencies such as Triton from a package index instead of building from source. However, the package index does not include triton_kernels.' 在 gfx950 上以 --enable-expert-parallel --tensor-parallel-size 4 运行 gpt-oss-120b 时，vLLM 默认附带的 triton_kernels 触发 'triton.runtime.errors.OutOfResources: out of resource: shared memory, Required: 165888, Hardware limit: 163840'。关联 issue #49778 报告了相同故障环境（ROCm 7.14.60850、MI355X gfx950、PyTorch 2.12.0+rocm7.14.0、Python 3.14.6），即 ROCm 7.14 引入 wheel 化 Triton 安装后 triton_kernels 缺失导致的回归。

实现拆解

实现按“构建层分派 → 运行层统一导入 → 容器层去重 → 验证配套”四步展开：

1. 构建层分派依赖来源（cmake/external_projects/triton_kernels.cmake）：删除单一版本变量 DEFAULT_TRITON_KERNELS_TAG，在 TRITON_KERNELS_SRC_DIR 未设置的分支下按 VLLM_TARGET_DEVICE STREQUAL "rocm" 分派——ROCm 使用 ROCm/triton fork（internal/3.6.x 线的固定 commit 0f380657dbf3ee86eb57558ff71df24f03b5d4e7），其余平台保持 triton-lang/triton 的 v3.5.1；FetchContent_Declare 的 GIT_REPOSITORY / GIT_TAG 改为引用变量。原因是 gfx950 需要 ROCm 分支中对共享内存做适配的 GPT-OSS 内核。
2. 运行层统一导入契约（vllm/model_executor/layers/fused_moe/experts/gpt_oss_triton_kernels_moe.py）：_patch_make_bitmatrix_metadata() 去除 current_platform.is_rocm() 分流，原先“ROCm 导入外部 triton_kernels / 其他平台回退 vllm.third_party.triton_kernels”的两路导入统一为从外部 triton_kernels.tensor_details 导入 _bm、BitmatrixMetadata、_keyed_add、cdiv、sum_bitmatrix_rows；ImportError

时静默返回。原因是构建期 CMake 已在各平台统一安装外部包，vendored 回退成为死路径，统一后能保证 ROCm 构建实际使用 CMake 拉取的 ROCm 版本。

3. 容器构建去重（docker/Dockerfile.rocm_base）：删除 RUN if [-d triton/python/triton_kernels]; then pip install build && cd triton/python/triton_kernels && python3 -m build --wheel ... 这一手工 wheel 构建层，避免与 CMake 安装路径重复、两条路径并存导致版本漂移。
4. 测试与配套：本次未新增测试文件，验证完全依赖现有 CI 门禁与 PR body 列出的手工用例——CI 组包括 mi355_1 Entrypoints Integration（API Server Generate / OpenAI Part 2）、mi355_2 GPQA Eval（GPT-OSS）、mi355_1 V1 Spec Decode，隔离用例为 entrypoints/tool_parsers/test_openai_tool_parser.py::test_calculator_tool_call_and_argument_accuracy，PR body 报告在 gfx950 + ROCm 7.14 上全部通过。提交演进上还经历了“Don't gate on ROCm version”（取消版本门控）与“Remove is_rocm guard & simplify import”（消除平台分流）两次收敛。

关键文件：

- vllm/model_executor/layers/fused_moe/experts/gpt_oss_triton_kernels_moe.py（模块 内核适配；类别 source；类型 data-contract；符号 _patch_make_bitmatrix_metadata）：运行时导入契约变更：_patch_make_bitmatrix_metadata() 移除 current_platform.is_rocm() 分流与 vendored 回退导入，所有平台统一从外部 triton_kernels 包导入，确保 CMake 安装的 ROCm 版本被实际使用，是本次修复在运行时的落点。
- cmake/external_projects/triton_kernels.cmake（模块 构建脚本；类别 config；类型 core-logic）：本 PR 的核心修复点：按 VLLM_TARGET_DEVICE 分派 triton_kernels 的 FetchContent 来源，ROCm 平台改用 ROCm/triton fork 的固定 commit（internal/3.6.x 线），这是 gfx950 共享内存超限问题的真正解法。
- docker/Dockerfile.rocm_base（模块 Docker 镜像；类别 infra；类型 infrastructure）：删除在 Triton 源码树中手工构建 triton_kernels wheel 的 RUN 层，避免与 CMake 安装路径重复，保证依赖只有单一安装来源，与 ROCm 7.14 wheel 化方向一致。

关键符号：_patch_make_bitmatrix_metadata

关键源码片段

vllm/model_executor/layers/fused_moe/experts/gpt_oss_triton_kernels_moe.py

运行时导入契约变更：_patch_make_bitmatrix_metadata() 移除 current_platform.is_rocm() 分流与 vendored 回退导入，所有平台统一从外部 triton_kernels 包导入，确保 CMake 安装的 ROCm 版本被实际使用，是本次修复在运行时的落点。

```
def _patch_make_bitmatrix_metadata() -> None:
    """Monkey-patch make_bitmatrix_metadata to support non-power-of-2 top_k.

    triton's tl.arange requires a power-of-2 range. The original kernel
    computes BLOCK_SIZE = BLOCK_PER_TOK * TOKS_PER_ROW (= 32 * top_k). For
```

```
    DeepSeek-V4 with top_k=6 this gives 192, which is not a power of 2 and
    causes a compile error at the first forward pass.

    Fix: define a drop-in replacement kernel that accepts an extra constexpr
    BLOCK_SIZE_PADDED (next power of 2 >= BLOCK_SIZE) and uses it for the
    tl.arange call while keeping the actual BLOCK_SIZE as the stride between
    thread-blocks so that all flat indices into NonzeroIndx stay correct.
    Elements beyond BLOCK_SIZE are masked out (col_indx = 0xffff) and ignored.
    """
    import torch
    import triton
    import triton.language as tl

    try:
        # 变更前按平台分流：ROCm 导入外部 triton_kernels，其余平台回退到
        # vllm.third_party.triton_kernels 内置副本。现在构建期 CMake 已统一
        # 安装外部包，因此去掉 is_rocm 分支，所有平台行为一致，并确保 ROCm
        # 运行时使用的正是 CMake 拉取的 ROCm 版 triton_kernels。
        from triton_kernels.tensor_details import bitmatrix as _bm
        from triton_kernels.tensor_details.bitmatrix import (
            BitmatrixMetadata,
            _keyed_add,
            cdiv,
        )
        from triton_kernels.tensor_details.bitmatrix_details.sum_bitmatrix_rows import (
            sum_bitmatrix_rows, # noqa: E501
        )
    except ImportError:
        # 外部包缺失时静默跳过；此后 SparseMatrix 首次 forward 才会遇到
        # 非 2 的幂 block 的编译错误，排障时需注意这一点。
        return
```

cmake/external_projects/triton_kernels.cmake

本 PR 的核心修复点：按 VLLM_TARGET_DEVICE 分派 triton_kernels 的 FetchContent 来源，ROCm 平台改用 ROCm/triton fork 的固定 commit（internal/3.6.x 线），这是 gfx950 共享内存超限问题的真正解法。

```
  else()
    # 按目标设备选择 triton_kernels 的来源仓库与版本。
    # ROCm 7.14 之后 Triton 改从 AMD 包索引 wheel 安装，wheel 内不含
    # triton_kernels；且上游 v3.5.1 的 GPT-OSS 内核在 gfx950 上会触发
    # OutOfResources（shared memory 超限），因此必须改用 ROCm 分支版本。
    if (VLLM_TARGET_DEVICE STREQUAL "rocm")
      set(TRITON_GIT "https://github.com/ROCm/triton.git")
      # Pinned from release/internal/3.6.x —— 固定 commit 而非可变分支，
      # 避免 branch 被 force-push 导致任意代码进入构建（供应链固定）。
      set(TRITON_KERNELS_TAG "0f380657dbf3ee86eb57558ff71df24f03b5d4e7")
    else()
      set(TRITON_GIT "https://github.com/triton-lang/triton.git")
```

```
    set(TRITON_KERNELS_TAG "v3.5.1")
  endif()
  message (STATUS "[triton_kernels] Fetch from ${TRITON_GIT}:${TRITON_KERNELS_TAG}")

  FetchContent_Declare(
    triton_kernels
    # TODO (varun): 只从 Triton 仓库拉取 triton_kernels 子目录
    GIT_REPOSITORY ${TRITON_GIT}
    GIT_TAG ${TRITON_KERNELS_TAG}
    GIT_PROGRESS TRUE
    SOURCE_SUBDIR python/triton_kernels/triton_kernels
  )
```

评论区精华

唯一的 review 评论来自 depthfirst-app[bot]（LOW 严重度）：质疑 triton_kernels.cmake 中把可变分支 release/internal/3.6.x 用作 GIT_TAG 存在供应链固定缺口——该目录下其他 FetchContent 依赖均固定到 commit hash 或不可变 tag，分支可被 force-push 或静默更新导致任意代码进入构建；并给出替换为当前 tip commit 0f380657dbf3ee86eb57558ff71df24f03b5d4e7 的建议。该建议由作者以提交 8c50fa6（co-authored-by: depthfirst-app[bot]）直接落实，最终文件中 TRITON_KERNELS_TAG 即该 hash，并保留注释 '# Pinned from release/internal/3.6.x' 便于后续追溯。三位人类 reviewer（micah-wil、Rohan138、AndreasKaratzas）均 APPROVED；claude[bot] 因 PR 来自 fork 未执行自动 review。

- ROCm triton 依赖应固定到 commit 而非可变分支 (security): 已采纳：提交 8c50fa6（与 depthfirst-app[bot] 共同署名）将 TRITON_KERNELS_TAG 固定为 commit hash 0f380657dbf3ee86eb57558ff71df24f03b5d4e7，并保留注释 '# Pinned from release/internal/3.6.x' 便于后续更新。

风险与影响

- 风险：
  1. 非 ROCm 平台行为变更：原先 CUDA 平台导入 vendored vllm.third_party.triton_kernels，现在一律导入外部 triton_kernels 包。标准 CMake 构建无影响（本就会安装 v3.5.1），但绕过 CMake 的安装方式（如纯 pip 编译、源码直接运行）会触发 except ImportError 静默跳过，使 bitmatrix 非 2 的幂 top_k 补丁失效，问题延迟到首次 forward 才以编译错误暴露，排障成本高。
  2. 静默失败路径：_patch_make_bitmatrix_metadata 的 except ImportError: return 不产生任何告警，若外部包版本与补丁目标（sum_bitmatrix_rows、SparseMatrix.__post_init__依赖的内部结构）不匹配，失败不可见。
  3. ROCm 固定 commit 的维护负担：0f380657... 来自 AMD 内部 release/internal/3.6.x 线，ROCm 侧 Triton 持续演进时需 AMD 团队跟进重新固定与回归验证，与 #50307 Helion 升级类似，这类 pin 需要周期审计。
  4. 平台覆盖面有限：CMake 分支对所有 ROCm 设备统一使用该 fork 版本，但 PR 仅在 gfx950 验证；若 fork 内核在其他 gfx 架构（gfx90a、gfx942）与上游行为有差异，可

能引入回归。

5. Docker 隐含假设：Dockerfile.rocm_base 删除手工安装层后，隐含假设 CMake 一定会安装 triton_kernels；若有镜像构建路径未执行该 CMake 目标，镜像内将缺失该包且无兜底。 - 影响：用户侧：修复 issue #49778 报告的 gpt-oss-120b 在 ROCm 7.14 + gfx950（MI355X）上的 OutOfResources 崩溃，受影响 CI 组（Entrypoints Integration、GPQA Eval GPT-OSS、V1 Spec Decode）恢复通过；影响面限定在 ROCm 平台与使用 triton_kernels GPT-OSS 内核的模型（gpt-oss 系列及依赖 bitmatrix 的路径）。构建侧：ROCm 构建新增一个外部 FetchContent 依赖（ROCm/triton fork），首次构建需拉取新仓库；Docker ROCm 基础镜像减少一个 wheel 构建步骤，构建时间缩短且安装路径单一化。团队侧：AMD 需要维护 fork commit 的更新与验证；固定 commit 同时提升了构建可复现性。整体影响为中等、局部，不改动推理主路径逻辑。 - 风险标记：构建依赖分平台分派，fork 版本需持续维护，except ImportError 静默吞掉，补丁失效无告警，非 ROCm 平台移除 vendored 回退，仅在 gfx950 验证，其他 gfx 架构未覆盖

关联脉络

- PR #50516 [ROCm][CI] Fall back to lossless Kimi K3 MXFP4 emulation on gfx942: 同为 gfx9xx 平台上的 ROCm 内核回退 / 适配修复，且涉及 fused_moe 相关路径，可与本 PR 对照 gfx942 与 gfx950 两类 AMD Instinct 平台的内核适配策略。
- PR #50307 Bump Helion to 1.4.0: 同为 ROCm 构建 / 内核修复链路：升级 ROCm 构建依赖并修复内核数值问题，与本 PR 的分平台依赖固定属于同一 ROCm 支持演进线。
- PR #48949 [ROCm][Quark][7/N] Use MXFP4 linear kernel abstraction for emulation backend: ROCm 内核选择与抽象重构（MXFP4 统一内核选择），与本 PR 统一 triton_kernels 导入路径的思路一致，体现 ROCm 支持线向共享化收敛的趋势。