PR #41946 完整报告

vllm-project/vllm

[Bugfix] [ROCm] [DSV4] [Perf] Add aiter mhc support

合并时间：2026-05-13 21:43

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/41946

执行摘要

- 一句话：集成 AITER 的 mHC 内核，优化 ROCm 上 DeepSeek-V4 推理性能并修复路径问题
- 推荐动作：建议仔细阅读本 PR，特别是 CustomOp 的分派模式、_tilelang_ops.py 的懒加载设计以及 _forward_rocm 与 _forward_cuda 的分离。这些设计决策对维护多后端 kernel 具有参考价值。对于性能敏感场景，应跟踪 AITER 新版本以移除当前 workaround。

功能与动机

根据 PR 描述，主要动机包括：1）优化 ROCm 上的 aiter mHC 内核，提升 DeepSeek-V4 推理性能；2）修复因 PR#41536 导致的 ROCm 路径损坏（关键 bug）；3）移除对不支持 tilelang 平台的依赖，通过懒加载避免导入错误。

实现拆解

1. 重构 MHC Kernel 架构：将原本全部实现在 vllm/model_executor/layers/mhc.py 中的 tilelang kernel 分离到 vllm/model_executor/kernels/mhc/ 下的 tilelang.py、triton.py、aiter.py、torch.py 文件中。使用 CustomOp 类分派 forward_cuda / forward_hip 路径，简化主文件。
2. 添加 AITER 内核：在 vllm/model_executor/kernels/mhc/aiter.py 中封装 mhc_pre_aiter 和 mhc_post_aiter，调用 rocm_aiter_ops.mhc_pre/post。同时在 vllm/_aiter_ops.py 中添加对应 AITER C++/HIP 操作的 Python 绑定，包含 hc_head 等。
3. TileLang 懒加载与平台守卫：在 vllm/_tilelang_ops.py 中定义 compute_num_split、mhc_pre_big_fuse_tilelang 等 tilelang JIT 函数。该文件仅在 current_platform.is_cuda() 为真时加载 tilelang，否则置为 None。所有引用 tilelang 的模块（如 tilelang.py）通过函数内部导入 _tilelang_ops，实现按需加载。
4. 模型适配：修改 deepseek_v4.py 的 MHC block，增加 _forward_rocm 路径，根据平台决定调用 _forward_cuda 或 _forward_rocm。同时将 torch.ops.vllm.mhc_pre 调用替换为 MHCPreOp 等 CustomOp 实例，统一分发。MTP 模块做了相应调整。
5. 测试与验证：在 tests/kernels/test_mhc_kernels.py 中添加 test_hc_head_triton 单元测试，覆盖多种参数组合（不同 hc_mult 和 hidden_size），全部通过。在 ROCm 硬件（mi355x）上运行 DeepSeek-V4 Pro lm-eval，确认正确性。

关键文件：

- vllm/model_executor/kernels/mhc/tilelang.py（模块 内核实现；类别 source；类型 data-contract；符号 mhc_pre_tilelang, _mhc_pre_tilelang_fake, mhc_post_tilelang, mhc_fused_post_pre_tilelang）：新增 tilelang 后端封装，提供 mHC pre、post、fused 等操作，是 CUDA 路径的核心实现。通过 direct_register_custom_op 注册自定义 op。
- vllm/model_executor/kernels/mhc/triton.py（模块 内核实现；类别 source；类型 data-contract；符号 _rmsnorm_nw_kernel, rmsnorm_nw, _hc_head_reduce_store_kernel, hc_head_reduce_triton_kernel）：新增 Triton 后端实现，包含无权重 RMSNorm 和 hc_head reduce 内核，用于 ROCm 和 CUDA 的备用路径。
- vllm/model_executor/kernels/mhc/aiter.py（模块 内核实现；类别 source；类型 data-contract；符号 mhc_pre_aiter, _mhc_pre_aiter_fake, mhc_post_aiter, _mhc_post_aiter_fake）：新增 AITER 后端封装，将 ROCm 优化内核集成到 mHC 流程，是 ROCm 性能提升关键。
- vllm/_tilelang_ops.py（模块 基础设施；类别 source；类型 dependency-wiring；符号 compute_num_split, mhc_pre_big_fuse_tilelang, mhc_fused_tilelang, mhc_post_tilelang）：全局 tilelang 操作定义，包含 split-k 计算和多个 tilelang JIT kernel，是懒加载的入口。
- vllm/model_executor/layers/mhc.py（模块 模型层；类别 source；类型 data-contract；符号 compute_num_split, mhc_pre_big_fuse_tilelang, MHCPreOp, enabled）：主层文件，大幅重构：移除内联 tilelang 代码，改为导入 kernels 目录下的模块，使用 CustomOp 进行硬件分派。
- vllm/model_executor/models/deepseek_v4.py（模块 模型实现；类别 source；类型 data-contract；符号 forward, _forward_cuda, _forward_rocm, hc_head）：模型定义，增加 _forward_rocm 路径，根据平台分派 mHC 实现，引入 HCHeadOp 等 CustomOp。
- tests/kernels/test_mhc_kernels.py（模块 测试；类别 test；类型 test-coverage；符号 hc_head_ref, test_hc_head_triton）：新增 hc_head_triton 单元测试，覆盖多种参数组合，验证 Triton 内核的正确性。

关键符号：compute_num_split, mhc_pre_big_fuse_tilelang, mhc_pre_tilelang, mhc_post_tilelang, mhc_fused_post_pre_tilelang, mhc_pre_aiter, mhc_post_aiter, MHCPreOp, MHCPostOp, HCHeadOp, hc_head_triton, _forward_rocm

关键源码片段

vllm/model_executor/kernels/mhc/tilelang.py

新增 tilelang 后端封装，提供 mHC pre、post、fused 等操作，是 CUDA 路径的核心实现。通过 direct_register_custom_op 注册自定义 op。

```
# SPDX-License-Identifier: Apache-2.0
# SPDX-FileCopyrightText: Copyright contributors to the vLLM project
import torch
from vllm.utils.torch_utils import direct_register_custom_op

def mhc_pre_tilelang(
    residual: torch.Tensor,
```

```
    fn: torch.Tensor,
    hc_scale: torch.Tensor,
    hc_base: torch.Tensor,
    rms_eps: float,
    hc_pre_eps: float,
    hc_sinkhorn_eps: float,
    hc_post_mult_value: float,
    sinkhorn_repeat: int,
    n_splits: int = 1,
) -> tuple[torch.Tensor, torch.Tensor, torch.Tensor]:
    # 在函数内部懒加载 tilelang ops，避免模块级导入失败
    from vllm._tilelang_ops import compute_num_split, mhc_pre_big_fuse_tilelang
    from vllm.utils.deep_gemm import tf32_hc_prenorm_gemm
    from vllm.utils.math_utils import cdiv

    # 形状检查和计算
    assert residual.dtype == torch.bfloat16
    assert fn.dtype == torch.float32
    hc_mult = residual.shape[-2]
    hidden_size = residual.shape[-1]
    hc_mult3 = hc_mult * 2 + hc_mult * hc_mult
    hc_hidden_size = hc_mult * hidden_size
    outer_shape = residual.shape[:-2]
    residual_flat = residual.view(-1, hc_mult, hidden_size)
    num_tokens = residual_flat.shape[0]

    # 根据 grid size 计算 split-k 数量
    block_k, block_m = 64, 64
    n_splits = compute_num_split(block_k, hc_hidden_size, cdiv(num_tokens, block_m))

    # 预分配输出和中间张量
    post_mix = torch.empty(num_tokens, hc_mult, dtype=torch.float32, device=residual.device)
    comb_mix = torch.empty(num_tokens, hc_mult * hc_mult, dtype=torch.float32, device=
    residual.device)
    layer_input = torch.empty(num_tokens, hidden_size, dtype=torch.bfloat16, device=residual.
    device)
    gemm_out_mul = torch.empty(n_splits, num_tokens, hc_mult3, dtype=torch.float32, device=
    residual.device)
    gemm_out_sqrsum = torch.empty(n_splits, num_tokens, dtype=torch.float32, device=residual.
    device)

    # 调用 tf32 prenorm GEMM 计算乘法和平方和
    tf32_hc_prenorm_gemm(
        residual_flat.view(num_tokens, hc_mult * hidden_size),
        fn, gemm_out_mul, gemm_out_sqrsum, n_splits,
    )

    # tilelang 融合 kernel 完成后续归一化、sinkhorn 等
    mhc_pre_big_fuse_tilelang(
```

```
        gemm_out_mul, gemm_out_sqrsum, hc_scale, hc_base, residual_flat,
        post_mix, comb_mix, layer_input,
        hidden_size, rms_eps, hc_pre_eps, hc_sinkhorn_eps,
        hc_post_mult_value, sinkhorn_repeat, n_splits, hc_mult,
    )

    return (
        post_mix.view(*outer_shape, hc_mult, 1),
        comb_mix.view(*outer_shape, hc_mult, hc_mult),
        layer_input.view(*outer_shape, hidden_size),
    )
```

评论区精华

在 code review 中，gemini-code-assist[bot] 指出了几处性能问题：torch.device 上下文管理器在热路径上的开销、每次 forward 都重新分配和复制权重的低效、以及 mhc_post 中 torch.empty_like 的分配开销。tjtanaa 回应称 torch.device 是 AITER v0.1.13 所需，需要等待 AITER 新版本解决；而权重预处理的问题因 CustomOp 不管理权重而暂未处理，未来可考虑在 process_weights_after_loading 中优化。gnovack 询问将 tilelang kernel 放在根目录 _tilelang_ops.py 的原因，tjtanaa 解释了懒加载和避免平台相关条件定义，gnovack 表示同意该方案。

- AITER mhc_pre 中 torch.device 上下文管理器的性能开销 (performance): tjtanaa 回复称在当前 AITER v0.1.13 版本中无法移除，否则会导致 OOM；需要等待 AITER PR#2916 修复，后续升级时将解决。
- 每次 forward 都重新分配和复制权重的低效问题 (performance): 未直接回复，但 PR 已合并，该问题待后续优化。
- tilelang kernels 文件放置位置的设计讨论 (design): 决议保留现有设计，在根目录 _tilelang_ops.py 中放置 tilelang 相关操作。

风险与影响

- 风险： 1) AITER 版本依赖：当前实现基于 AITER v0.1.13，其 torch.device 和内存分配问题可能影响 ROCm 性能，需在升级后移除这些 workaround。 2) TileLang 懒加载条件：_tilelang_ops.py 通过 current_platform.is_cuda() 判断是否加载 tilelang，可能在某些混合平台上误判。 3) 权重预计算缺失：hc_head 中每次 forward 都复制和 pad 权重，增加了不必要的开销。 4) 热路径分配：mhc_post 等操作中直接 torch.empty_like 分配输出缓冲区，可能触发 GPU 同步和碎片化，影响推理延迟。
  - 影响：用户：ROCm 平台使用 DeepSeek-V4 将获得显著性能提升（经 lm-eval 验证），同时修复了之前的 ROCm 崩溃问题。系统：kernel 文件按后端分离，使代码结构和模块化更好。团队：为后续 AITER 版本升级和进一步优化奠定了基础。预计对 NVIDIA 用户无负面影响（CUDA 路径保持不变）。
- 风险标记：ROCm 特定路径，AITER 版本依赖，热路径性能风险，权重预处理缺失

关联脉络

- PR #41536 [Bugfix][ROCm] (from PR body description, title unknown): 本 PR 明确指出修复了该 PR 导致的 ROCm 路径损坏。