PR #43679 完整报告

vllm-project/vllm

[ROCm][DSV4] Enable Tilelang MHC replacing torch/triton mhc

合并时间：2026-05-28 15:05

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/43679

执行摘要

- 一句话：Tilelang MHC 替换 Torch/Triton 并支持 ROCm
- 推荐动作：建议阅读范围：所有涉及 DeepSeek V4 推理优化、ROCm 支持、TileLang kernel 集成的工程师。 关注点：
- _tilelang_ops.py 中平台条件编译和 PDL 设计，为跨平台 kernel 提供参考。
- mhc.py 中 HAS_TILELANG 调度模式，体现优雅降级策略。
- Review 中关于 warp size 和 eager CUDA 初始化的讨论，了解跨平台 kernel 常见陷阱。
- 测试文件 test_mhc_kernels.py 覆盖了 tilelang 和 torch 双路径验证，值得作为类似 PR 的测试模板。

功能与动机

Tilelang 最新版本支持 vendor-free 编译（CUDA/ROCm），参考 sglang 实现添加 ENABLE_PDL 控制以在不支持 PDL 的平台（如 ROCm）上禁用。目的是用更快的 tilelang 内核替换慢速的 torch/triton 内核，提升 ROCm 上的推理性能，同时保留 torch 回退路径以兼容无 tilelang 环境，并为后续 aiter MHC 对比提供基准。

实现拆解

实现步骤

1. 平台 PDL 支持检测：在 vllm/platforms/cuda.py 新增 is_arch_support_pdl 类方法，通过计算能力判断硬件是否支持 PDL（≥9）。在 vllm/platforms/interface.py 添加抽象方法。
2. TileLang 内核注册：在 vllm/_tilelang_ops.py 中定义 ENABLE_PDL 全局变量，将 PDL sync/trigger 包装在条件中；新增 hc_prenorm_gemm_tilelang 和 hc_prenorm_gemm_block_m_tilelang 两个 tilelang kernel，用于计算 GEMM 与平方和。保留原有 mhc_pre_big_fuse_tilelang 等函数并根据平台调整 pass_configs。
3. MHC 层调度重构：在 vllm/model_executor/layers/mhc.py 中添加 HAS_TILELANG 标志；forward_hip 方法首先检查此标志，若为真则调用 torch.ops.vllm.mhc_pre_tilelang 等操作，否则调用 forward_native 执行 torch 实现。forward_native 从原先的 raise 改为实际实现，确保回退路径可用。类似改动也应用于 MHCPostOp 和 HCHeadOp。
4. GEMM 双路径选择：在 vllm/model_executor/kernels/mhc/tilelang.py 中新增纯 torch 实现的 _torch_hc_prenorm_gemm 和 tilelang 实现的 _tilelang_hc_prenorm_gemm；mhc_pre_tilelang 函数根据 is_deep_gemm_supported() 选择使用 tf32 还是 tilelang

gemm。

5. 模型 forward 路径适配：在 vllm/models/deepseek_v4/amd/model.py 中，根据 self.has_tilelang 选择调用 _forward_fused_post_pre（tilelang 可用）或 _forward_unfused_post_pre（torch 回退）；同时删除依赖平台判断的 _forward_cuda/_forward_rocm，改用统一的 fused/unfused 命名。
6. 测试与依赖：在 tests/kernels/test_mhc_kernels.py 增加 tilelang 分支测试，覆盖不同 token 数和 hidden_size；更新 requirements/test/rocm.in 和 requirements/test/rocm.txt，添加 tilelang>=0.1.10 依赖并最终固定版本。

关键文件：

- vllm/_tilelang_ops.py（模块 TileLang 内核；类别 source；类型 core-logic；符号 hc_prenorm_gemm_tilelang, hc_prenorm_gemm_block_m_tilelang, mhc_pre_big_fuse_tilelang, mhc_pre_big_fuse_with_norm_tilelang）：核心 TileLang 内核注册与 PDL 控制，平台差异化编译的关键文件。
- vllm/model_executor/kernels/mhc/tilelang.py（模块 MHC 内核；类别 source；类型 core-logic；符号 _torch_hc_prenorm_gemm, _tilelang_hc_prenorm_gemm, mhc_pre_tilelang）：新增 torch 和 tilelang 两种 prenorm gemm 实现，并由上层根据 deep_gemm 支持选择路径。
- vllm/model_executor/layers/mhc.py（模块 MHC 层；类别 source；类型 data-contract；符号 forward_native, forward_hip, forward_cuda, HCHeadOp.forward_hip）：MHC 调度层引入 HAS_TILELANG 标志并优先使用 tilelang 操作，同时提供完整的 torch 回退实现。
- vllm/models/deepseek_v4/amd/model.py（模块 DeepSeekV4 模型；类别 source；类型 data-contract；符号 _forward_fused_post_pre, _forward_unfused_post_pre, forward）：模型 forward 路径根据 tilelang 可用性选择 fused/unfused 策略，抽象化平台差异。
- vllm/platforms/cuda.py（模块 平台识别；类别 source；类型 core-logic；符号 is_arch_support_pdl）：添加 is_arch_support_pdl 硬件检测方法，用于控制 PDL 启用。
- tests/kernels/test_mhc_kernels.py（模块 MHC 测试；类别 test；类型 test-coverage；符号 test_mhc_pre_tilelang, test_hc_prenorm_gemm_tilelang, test_mhc_post_tilelang, test_hc_head_tilelang）：扩展 tilelang 分支的测试用例，验证双平台正确性。
- vllm/platforms/interface.py（模块 平台接口；类别 source；类型 core-logic；符号 is_arch_support_pdl）：在平台接口基类中添加 is_arch_support_pdl 抽象方法，确保子类实现。
- requirements/test/rocm.txt（模块 依赖配置；类别 docs；类型 documentation）：添加 tilelang 依赖并固定版本。

关键符号：is_arch_support_pdl, hc_prenorm_gemm_tilelang, hc_prenorm_gemm_block_m_tilelang, _torch_hc_prenorm_gemm, _tilelang_hc_prenorm_gemm, _forward_fused_post_pre, _forward_unfused_post_pre

关键源码片段

vllm/_tilelang_ops.py

核心 TileLang 内核注册与 PDL 控制，平台差异化编译的关键文件。

```
# SPDX-License-Identifier: Apache-2.0
import math
from functools import cache
from typing import TYPE_CHECKING, Any
import torch
from vllm.platforms import current_platform
from vllm.utils.import_utils import has_tilelang
from vllm.utils.math_utils import cdiv

# TileLang 用于 MHC，兼容 CUDA 和 ROCm
if TYPE_CHECKING or current_platform.is_cuda_alike():
    if not has_tilelang():
        raise ImportError(
            "tilelang is required for mhc but is not installed. Install it with "
            "`pip install tilelang`."
        )
    import tilelang
    import tilelang.language as T
else:
    tilelang = None
    T = None

# 仅在 CUDA 且计算能力 >=9 时启用 PDL，ROCm 上禁用
ENABLE_PDL = current_platform.is_arch_support_pdl() and current_platform.is_cuda()

# 基础 pass_configs，CUDA 额外设置 PTX 寄存器使用级别
pass_configs: dict[tilelang.PassConfigKey, Any] = {
    tilelang.PassConfigKey.TL_DISABLE_WARP_SPECIALIZED: True,
    tilelang.PassConfigKey.TL_DISABLE_TMA_LOWER: True,
}
if current_platform.is_cuda():
    pass_configs[tilelang.PassConfigKey.TL_PTXAS_REGISTER_USAGE_LEVEL] = 10

@tilelang.jit(pass_configs=pass_configs)
def mhc_pre_big_fuse_tilelang(...):
    # ...
    with T.Kernel(num_tokens, threads=96) as i:
        if ENABLE_PDL:
            T.pdl_sync()
        # ... 计算逻辑 ...
        if ENABLE_PDL:
            T.pdl_trigger()
```

vllm/model_executor/kernels/mhc/tilelang.py

新增 torch 和 tilelang 两种 prenorm gemm 实现，并由上层根据 deep_gemm 支持选择路径。

```
def _tilelang_hc_prenorm_gemm(
    x: torch.Tensor,
    fn: torch.Tensor,
```

```
    out: torch.Tensor,
    sqrsum: torch.Tensor,
    hidden_size: int,
    hc_mult: int,
    tile_n: int = 12,
    n_thr: int = 512,
    n_splits: int = 1,
) -> None:
    """
    TileLang 实现的 prenorm GEMM，替代 tf32 版本，
    支持 CUDA 和 ROCm 双平台。
    """
    from vllm._tilelang_ops import (
        hc_prenorm_gemm_block_m_tilelang,
        hc_prenorm_gemm_tilelang,
    )
    # 尺寸断言
    assert out.shape[0] == n_splits
    assert sqrsum.shape[0] == n_splits
    assert x.shape[1] == hc_mult * hidden_size
    assert x.shape[1] % n_splits == 0
    assert (x.shape[1] // n_splits) % n_thr == 0

    use_default_config = tile_n == 12 and n_thr == 512

    # 大 batch 使用 block_m kernel 减少开销
    if n_splits == 1 and use_default_config and x.shape[0] >= 1024:
        hc_prenorm_gemm_block_m_tilelang(
            x, fn, out, sqrsum, hidden_size, hc_mult,
            fn.shape[0], n_thr, tile_n, 2,
        )
        return

    # 小 batch 且 hidden size 对齐时调整 tile_n
    if (n_splits == 1 and use_default_config
            and x.shape[0] < 128 and x.shape[1] % 1024 == 0):
        hc_prenorm_gemm_tilelang(
            x, fn, out, sqrsum, hidden_size, hc_mult,
            fn.shape[0], 1024, 4, n_splits,
        )
        return

    # 通用路径
    hc_prenorm_gemm_tilelang(
        x, fn, out, sqrsum, hidden_size, hc_mult,
        fn.shape[0], n_thr, tile_n, n_splits,
    )
```

vllm/model_executor/layers/mhc.py

MHC 调度层引入 HAS_TILELANG 标志并优先使用 tilelang 操作，同时提供完整的 torch 回退实现。

```
from vllm.utils.import_utils import has_tilelang

HAS_TILELANG = has_tilelang()

@CustomOp.register("mhc_pre")
class MHCPreOp(CustomOp):
    # ...
    def forward_hip(self, residual, fn, hc_scale, hc_base,
                    rms_eps, hc_pre_eps, hc_sinkhorn_eps,
                    hc_post_mult_value, sinkhorn_repeat,
                    n_splits=1, norm_weight=None, norm_eps=0.0):
        # 优先使用 tilelang 内核
        if HAS_TILELANG:
            return torch.ops.vllm.mhc_pre_tilelang(
                residual, fn, hc_scale, hc_base,
                rms_eps, hc_pre_eps, hc_sinkhorn_eps,
                hc_post_mult_value, sinkhorn_repeat,
                n_splits, norm_weight, norm_eps,
            )
        else:
            return self.forward_native(
                residual, fn, hc_scale, hc_base,
                rms_eps, hc_pre_eps, hc_sinkhorn_eps,
                hc_post_mult_value, sinkhorn_repeat,
                n_splits, norm_weight, norm_eps,
            )

    def forward_native(self, residual, fn, hc_scale, hc_base,
                       rms_eps, hc_pre_eps, hc_sinkhorn_eps,
                       hc_post_mult_value, sinkhorn_repeat,
                       n_splits=1, norm_weight=None, norm_eps=0.0):
        # torch 回退实现，调用原来的 mhc_pre_torch
        return mhc_kernels.mhc_pre_torch(
            residual, fn, hc_scale, hc_base,
            rms_eps, hc_pre_eps, hc_sinkhorn_eps,
            hc_post_mult_value, sinkhorn_repeat,
        )
```

评论区精华

Review 讨论点

- Warp Size 硬编码：gemini-code-assist 指出 hard-coded 32 在 ROCm 上可能不正确（wavefront 64）。tjtanaa 回应称 TileLang 的 HIP reduce 实现假定逻辑 warp 大小 32（如 __shfl_xor(value, 16, 32)），因此保持 32 是安全的。该 decision 被接受，未修改。

- Eager CUDA 初始化：gemini-code-assist 建议 is_arch_support_pdl 使用 cls.get_device_capability(0) 避免 torch.cuda.current_device() 导致的 eager CUDA 初始化。tjtanaa 认为单一节点内 GPU 同构，仅触发一次无影响，且参考了 sglang 的实现方式。该建议未被采纳。
- TileLang 版本固定：AndreasKaratzas 建议固定 tilelang 版本以防未来回归。tjtanaa 先在评论中同意后续跟进，后在最后一次提交中固定为 0.1.10。
- Compressor 导入错误：tjtanaa 发现并临时修复了 deepseek_v4/compressor.py 的 cutlass 未找到错误，但 WoosukKwon 指出已在 PR #43710 中单独修复，tjtanaa 移除了自己的修改。
  - Warp size 硬编码问题 (correctness): tjtanaa 回应 TileLang 的 HIP reduce 使用逻辑 32-lane warp，32 是安全的，未修改。
  - Eager CUDA 初始化 (performance): tjtanaa 认为单一节点同构 GPU 无影响，参考 sglang 实现，未修改。
  - TileLang 版本固定 (documentation): tjtanaa 同意并在后续提交中固定版本。
  - Compressor 导入错误修复 (other): 移除修改，依赖 #43710 修复。

风险与影响

- 风险：
  1. Warp Size 假设风险：TileLang 的 HIP reduce 当前使用 32-lane warp，如果未来版本改变，可能导致 ROCm 上结果错误。该风险目前通过文档注释和 review 讨论记录。
  2. Eager CUDA 初始化风险：is_arch_support_pdl 在模块导入时调用 torch.cuda.current_device()，可能干扰分布式初始化或 multiprocessing 环境。当前代码仅在 _tilelang_ops.py 导入时执行，且仅在 CUDA/ROCm 平台上，一般认为影响有限。
  3. TileLang 依赖风险：如果没有安装 tilelang 或安装版本不兼容，代码会回退到 torch 实现，功能正常但性能下降。测试覆盖了回退路径。
  4. PDL 控制逻辑：ENABLE_PDL 仅当 CUDA 且计算能力 ≥ 9 时启用，ROCm 上为 False，因此 pdl_sync 等操作跳过。如果未来 ROCm 支持 PDL，需要更新该条件。
  5. 性能回归风险：在 CUDA 上 tilelang 行为应与之前一致（之前 tilelang 已用于 CUDA），但切换为统一路径后可能引入细微差异。PR 测试数据显示无回归。
- 影响：影响范围：
  - 用户：ROCm 用户将获得显著的推理吞吐提升（PR 提供对比数据：无 MTP 场景下 TPOT 降低约 36.5%，吞吐提升 15.4%）。CUDA 用户体验无影响。
  - 系统：新增 tilelang 依赖，需在 ROCm 环境中安装。代码结构使 tilelang 与 torch 回退路径清晰分离。
  - 团队：为后续统一 MHC kernel 重构（@WoosukKwon 计划）奠定基础；tilelang 内核在 ROCm 上可用后，可进一步探索 aiter MHC 对比。
  - 影响程度：高，因为改变了 DeepSeek V4 模型在 ROCm 上的核心计算路径。
  - 风险标记：Warp size 硬编码假设，Eager CUDA 初始化，TileLang 版本依赖，PDL 控制仅限 CUDA

关联脉络

- PR #43710 [Bugfix] Fix cutlass import error in compressor: 修复了同一文件中因 #43584 引入的 cutlass 导入错误，tjtanaa 移除了自己的修复并依赖此 PR。