

PR #55991 完整报告

vllm-project/vllm

[ROCm][AITER] Skip AITER norm kernels when flattening to 2D would copy

合并时间: 2026-09-16 15:26

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/55991>

执行摘要

- 一句话: AITER norm 派发新增展平守卫, 避免隐藏整张量拷贝
- 推荐动作: 值得精读, 尤其是 ROCm 平台维护者、负责 IR op dispatch / 内核路由的开发者, 以及任何要写 AITER 或其他设备专用 op 谓词的人。三个最值得关注的设计点: 其一, 把「数学上支持」的谓词扩展成「数学支持且不做隐藏拷贝」的收益性判定——这是 `supports_args` 语义的一次细化, 改变了谓词作为内核选择裁决层的定位; 其二, `flatten_to_2d_is_free` 用 `stride` 推导复现 `Tensor.reshape` 的视图可表达性, 并特意豁免 `size==1` 的维, 是可直接复用的工具函数写法; 其三, 作者主动披露 `vllm_c` 兜底缺口并拒绝在本 PR 扩大范围, 这种「精确收益 + 已知缺口」的分割方式值得借鉴。阅读时建议重点核对三件事: 谓词拒绝后回退路径在当前版本 Inductor 下是否真的不物化拷贝、`-O.backend=eager` 等关闭 Inductor 的配置是否会出现小幅回退、以及本 PR 未提供自动化 dispatch 断言所带来的长期回归风险。

功能与动机

PR body 说明: AITER norm 内核只接受 2-D 输入, `vllm/kernels/aiter_ops.py` 的包装层用 `x.reshape(-1, x_original_shape[-1])` 展平前导维, 而 `reshape` 在展平形状无法用输入 `strides` 表达时会静默退化为 `.contiguous()`, 注入一次整张量 device copy; 原有谓词只检查 `dtype` 与 `variance_size`, 因此仍会选中 AITER, 「the copy costs more than the kernel saves」。典型受害路径是 Qwen3 系的 per-head QK norm: q/k 是 fused QKV 投影最后一维的切片, 视图为 `(num_tokens, num_heads, head_dim)`、stride 为 `(6144, 128, 1)`, 每一层都要付出 4096x4096 bf16 的拷贝。作者明确目标是在数值等价前提下「only the copy is gone」, 并强调谓词在 `VllmIRLoweringPass` 中按图节点求值一次, 不进 forward 稳态开销。

实现拆解

1. 新增几何判定函数 (`vllm/kernels/aiter_ops.py`): `flatten_to_2d_is_free(x)` 先对 `dim <= 2` 直接放行, 再要求 `x.stride(-1) == 1`, 随后从倒数第二维向前逐维检查 `x.size(i) != 1` and `x.stride(i) != expected_stride`, `expected_stride` 按右侧维尺寸累乘。这等价于复现 `Tensor.reshape` 的视图可表达性判定: 任一步长不匹配就意味着 `reshape` 会退化为 `contiguous()` 拷贝。size 为 1 的维被特意豁免, 避免把合法的广播 / 单头几何误判为需拷贝。
2. 把守卫接入两个注册谓词: `rms_no_var_16bit_only` 追加 and `flatten_to_2d_is_free(x)`; `rms_add_no_var_16bit_only` 同时检查 `x` 与 `x_residual` (融合 `add` 路径的两个输入都可能

带 strided 布局)。两处谓词文档字符串同步更新，说明新增的展平要求。谓词返回 False 时 `ir.ops.*.register_impl(supports_args=...)` 让 `IrOp.dispatch` 顺延到下一个 provider——ROCm 上即 native，Inductor codegen 直接消费任意 strides 并把 norm 融合进邻近算子。

3. 保持注册与内核包装层不变：`rms_norm / fused_add_rms_norm` 的 `impl` 函数体、`_rms_norm_impl` 中的 `flatten` 逻辑、`direct_register_aiter_op` 的 `custom op` 注册都未改动，改动集中在「何时允许 AITER 被选中」这一裁决层，因此不改变 kernel 实现本身与编译产物。
4. 测试配套 (`tests/kernels/ir/test_aiter_norm_dispatch.py`，新增 60 行)：辅助函数 `_qkv_slice_by_head` 复刻 fused-QKV 切片的真实几何；`test_flattenable` 覆盖 2-D/1-D/ 连续 3-D/ 含 `size==1` 维 / 最外层切片五种布局，`test_not_flattenable` 覆盖 QKV per-head 切片、最后一维步长非 1、中间维切片三种布局，并同时断言 `reshape(...).data_ptr()` 与 `x.data_ptr()` 的相等 / 不等关系，使谓词语义与 PyTorch 行为互相校验。
5. review 后的测试门控修正：`commit 03eebecb` 按 `tjtanaa` 意见加入 `pytestmark = pytest.mark.skipif(not is_aiter_found_and_supported(), ...)`，使该文件只在 ROCm 且 AITER 可用时运行，与仓库其他 AITER 测试模块保持一致。
6. 无 schema / 部署改动：全 Python 改动，不碰 C++/CUDA、不碰共享 `dispatch` 代码（作者在评论中确认 CUDA 与 RDNA4 因 `AITER_SUPPORTED=False` 与 `get_default_ir_op_priority` 不产出 `aiter` 而结构性不受影响）。

关键文件：

- `vllm/kernels/aiter_ops.py`（模块 AITER 内核；类别 source；类型 core-logic；符号 `flatten_to_2d_is_free`, `rms_no_var_16bit_only`, `rms_add_no_var_16bit_only`, `rms_norm`）：核心源码改动：新增 `flatten_to_2d_is_free` 并接入 `rms_no_var_16bit_only / rms_add_no_var_16bit_only` 两个 `supports` 谓词，直接决定 AITER norm 内核何时被 `IrOp.dispatch` 选中。
- `tests/kernels/ir/test_aiter_norm_dispatch.py`（模块 内核测试；类别 test；类型 test-coverage；符号 `_qkv_slice_by_head`, `test_flattenable`, `test_not_flattenable`）：新增测试：用 `_qkv_slice_by_head` 复刻 Qwen3 per-head QK norm 的真实几何，并断言谓词判定与 `reshape` 是否保留 `data_ptr` 严格一致，是验证守卫正确性的主要自动化手段。

关键符号：`flatten_to_2d_is_free`, `rms_no_var_16bit_only`, `rms_add_no_var_16bit_only`, `rms_norm`, `fused_add_rms_norm`, `_qkv_slice_by_head`, `test_flattenable`, `test_not_flattenable`

关键源码片段

`vllm/kernels/aiter_ops.py`

核心源码改动：新增 `flatten_to_2d_is_free` 并接入 `rms_no_var_16bit_only / rms_add_no_var_16bit_only` 两个 `supports` 谓词，直接决定 AITER norm 内核何时被 `IrOp.dispatch` 选中。

```
# vllm/kernels/aiter_ops.py
```

```
def flatten_to_2d_is_free(x: Tensor) -> bool:
```

```
"""Whether ``x.reshape(-1, x.shape[-1])`` is a view with unit-stride rows.
```

The AITER norm kernels only take dense 2D inputs, so the wrappers below flatten the leading dims with ``Tensor.reshape``, which silently falls back to ``contiguous()`` when the flattened shape is not expressible with the existing strides, adding a whole-tensor device copy that costs more than the kernel saves. A strided last dim flattens without a copy but not into a dense buffer, so it is rejected too.

```
"""
```

```
# 2-D 及以下不需要展平，必然不产生拷贝，直接放行
if x.dim() <= 2:
    return True
# 最后一维步长非 1 时，展平结果不是稠密缓冲，同样拒绝
if x.stride(-1) != 1:
    return False
# 从倒数第二维往前推：每一维步长必须等于其右侧所有维尺寸的乘积；
# 只要有一维不满足，前导维就无法合并成单维，reshape 会退化为 contiguous()
expected_stride = x.size(-1)
for i in range(x.dim() - 2, -1, -1):
    # size == 1 的维可忽略（任意步长都能被合并）
    if x.size(i) != 1 and x.stride(i) != expected_stride:
        return False
    expected_stride *= x.size(i)
return True
```

```
rms_no_var_16bit_only = (
    lambda x, weight, epsilon, variance_size=None: variance_size is None
    and x.dtype in (torch.float16, torch.bfloat16)
    and (weight is None or weight.dtype == x.dtype)
    # 新增条件：展平会拷贝时放弃 AITER，让 IrOp.dispatch 顺延到 native
    and flatten_to_2d_is_free(x)
)
"""AITER rms_norm only supports float16 and bfloat16 acts, no var_size override,
requires weight dtype to match x dtype, and requires flattening the activation
to 2D to be free."""
```

```
@ir.ops.rms_norm.register_impl(
    # supports_args 在 VllmIRLoweringPass 中按图节点求值一次，不进 forward 稳态
    "aiter", supports_args=rms_no_var_16bit_only, supported=AITER_SUPPORTED
)
def rms_norm(
    x: Tensor, weight: Tensor | None, epsilon: float, variance_size: int | None = None
) -> Tensor:
    # 内核包装层保持原样：__rms_norm_impl 里的 reshape 逻辑未改动，
    # 本 PR 只是让「会拷贝的布局」不再走到这里
    assert variance_size is None
    assert x.dtype in (torch.float16, torch.bfloat16)
```

```

if weight is None:
    weight = torch.ones(x.shape[-1], device=x.device, dtype=x.dtype)
return torch.ops.vllm_aiter.rms_norm(x, weight, epsilon)

```

```

rms_add_no_var_16bit_only = (
    lambda x, x_residual, weight, epsilon, variance_size=None: variance_size is None
    and x.dtype in (torch.float16, torch.bfloat16)
    and (weight is None or weight.dtype == x.dtype)
    # 融合 add 路径有两个输入，两者展平都必须免费
    and flatten_to_2d_is_free(x)
    and flatten_to_2d_is_free(x_residual)
)

```

tests/kernels/ir/test_aiter_norm_dispatch.py

新增测试：用 `_qkv_slice_by_head` 复刻 Qwen3 per-head QK norm 的真实几何，并断言谓词判定与 `reshape` 是否保留 `data_ptr` 严格一致，是验证守卫正确性的主要自动化手段。

```

# tests/kernels/ir/test_aiter_norm_dispatch.py

```

```

# review 后补上的门控：仅在 ROCm 且 AITER 可用时运行，与其他 AITER 测试模块一致
pytestmark = pytest.mark.skipif(
    not is_aiter_found_and_supported(),
    reason="Only test on ROCm with AITER installed and supported",
)

```

```

def _qkv_slice_by_head(num_tokens, num_q_heads, num_kv_heads, head_dim):
    """q viewed per-head from a fused QKV projection, as Qwen3-style QK-norm does."""
    # 复刻 fused QKV 布局：q 只是最后一维上的切片，再 view 成 (tokens, heads, head_dim);
    # 其 stride 为 (q_size + 2 * kv_size, head_dim, 1)，前导维无法合并 => reshape 必然拷贝
    q_size, kv_size = num_q_heads * head_dim, num_kv_heads * head_dim
    qkv = torch.empty(num_tokens, q_size + 2 * kv_size)
    q = qkv.split([q_size, kv_size, kv_size], dim=-1)[0]
    return q.view(*q.shape[:-1], num_q_heads, head_dim)

```

```

@pytest.mark.parametrize(
    "x",
    [
        torch.empty(8, 16), # 2-D，直接放行
        torch.empty(16), # 1-D
        torch.empty(2, 4, 16), # 连续 3-D，展平为视图
        torch.empty(2, 1, 16), # size == 1 的维可忽略
        torch.empty(8, 4, 16)[2:6], # 最外层切片仍保持连续
    ],
)
def test_flattenable(x):
    # 谓词必须与「reshape 是否保留 storage 指针」严格一致

```

```

assert flatten_to_2d_is_free(x)
assert x.reshape(-1, x.shape[-1]).data_ptr() == x.data_ptr()

@pytest.mark.parametrize(
    "x",
    [
        _qkv_slice_by_head(8, 32, 8, 128), # Qwen3 per-head QK norm 的真实几何
        torch.empty(4, 8, 32).transpose(-1, -2), # 最后一维步长非 1
        torch.empty(4, 8, 32)[: , :4], # 中间维切片，前导维无法合并
    ],
)
def test_not_flattenable(x):
    assert not flatten_to_2d_is_free(x)
    # reshape 必须拷贝，这正是谓词要挡掉的隐藏整张量开销
    assert x.reshape(-1, x.shape[-1]).data_ptr() != x.data_ptr()

```

评论区精华

PR 的讨论密度很高，主要围绕三条线：

1. 测试门控 (tjtanaa 的 inline review)：在 tests/kernels/ir/test_aiter_norm_dispatch.py: 22 处指出「We should skip this test file so with is_aiter_supported_and_available() (the function from vllm._aiter_ops)」，作者回复「Revised as suggested. Thanks.」并在 commit 03eebecb 中落地为 is_aiter_found_and_supported()。这是唯一一条 review comment，属于低争议的风格 / CI 兼容修正。
2. 数值精度追问：tjtanaa 要求补测精度，并给出关键判断——「AITER's common HIP path keeps normalization and weight multiplication in FP32 until the output store. Inductor can eliminate intermediate casts, so the actual difference depends on generated code as well as reduction order.」作者随后补了两份 gsm8k 结果：MI355X 上「No model moves more than 0.0061 strict-match, every delta is inside one standard error, and in 7 of 9 a second run of the unpatched build moves further than the patch does」；MI325X 上也给出 $\Delta \leq 0.0068$ 且 MoE 模型上 patch 的位移小于同一构建自比。tjtanaa 同时说明已手动触发小模型与大模型的 lmeval，覆盖面超出作者本地数据。
3. 作者自曝的已知缺口（未被本 PR 修复）：「the vllm_c fallback still copies」——当 Inductor 关闭（如 -O.backend=eager，优先级为 ["aiter", "vllm_c", "native"]）时，vllm/kernels/vllm_c.py 的 rms_no_var_size 接受任意 dtype 匹配输入并做同样的 flatten，本补丁会把 AITER 内核换成 vLLM 自己的内核但拷贝仍存在；fused_add_rms_norm 不受影响，因为 vllm_c 已把非连续输入交给 native。
4. 平台与稳态开销的论证：作者用「AITER_SUPPORTED 为 False 时 IrOp.dispatch 会 raise，所以 CUDA 优先级列表不可能含 aiter」+「CudaPlatform.get_default_ir_op_priority 从不输出 aiter」两条结构性理由排除 CUDA，用 not on_rdna4() 排除 RDNA4，并强调谓词只在 VllmIRLoweringPass 每节点求值一次、VLLM_ROCM_USE_AITER_RMSNORM 默认 True 使其默认生效。claude[bot] 的自动

review 因 PR 来自 fork 而被禁用，未产生额外技术意见。

- 测试文件缺少 AITER 可用性门控 (testing): 作者回复「Revised as suggested. Thanks.」, commit 03eebecb 加入 `pytestmark = pytest.mark.skipif(not is_aiter_found_and_should_supported())`, 与仓库其他 AITER 测试模块保持一致, 问题关闭。
- 数值精度回归: AITER HIP 路径与 Inductor 生成代码的差异 (correctness): 双方认定差异落在采样噪声与标准误范围内, tjtaana 另外手动触发小模型 / 大模型 lmeval 扩大覆盖面, 并以 LGTM 批准。
- vllm_c 兜底路径仍会拷贝的已知缺口 (design): 本 PR 不修复该缺口, 作者明确建议不要把 flattenability 检查扩展到 vllm_c, 仅作为已知限制记录, 留给后续工作。
- CUDA / RDNA4 与其他架构是否受影响 (design): 无需在 NVIDIA 硬件上实测, 影响面被限定在 ROCm + AITER + rank>2 strided slice; 结论基于代码路径推理加探针数据, 未在 CUDA 上运行验证。

风险与影响

- 风险:

1. **vllm_c** 兜底路径仍会拷贝 (已知且未修复): 当 Inductor 关闭、ROCm 优先级为 `["aiter", "vllm_c", "native"]` 时, `vllm/kernels/vllm_c.py` 的 `rms_no_var_size` 谓词不检查布局并在 `IS_ROCM and (x.dim() > 2 or not x.is_contiguous())` 分支做同样的 `reshape`, 本补丁实际是把更快的 AITER 内核换成 vLLM 内核、拷贝照旧, `rms_norm` 一条路径上可能出现轻微回退; `fused_add_rms_norm` 无此问题。作者在评论中显式承认并建议不要在本 PR 扩展修复范围。
2. 数值语义改变, 精度回归依赖抽样证据: `dispatch` 从 AITER HIP 内核 (FP32 归一化与权重乘法直到输出存储) 切到 Inductor 生成代码 (可消除中间 `cast`), 累加顺序与 `cast` 位置都变。gsm8k 全量 1319 题显示 `strict-match` 抖动 ≤ 0.0068 且在标准误内, 但仅覆盖 9 个模型与单一任务, 长链推理、代码生成或对数值极敏感的评测未覆盖; tjtaana 触发的 lmeval 是补充而非全量证明。
3. 承重假设依赖 Inductor 当前行为: 整条收益建立在「native 消费任意 strides 不物化拷贝」之上。作者用 kernel-level 数据支撑 (`strided slice` 只让 native 慢 18% / 3.2%, 而 `aiter` 慢 59% / 28%), 但这是特定 torch/Inductor 版本与 gfx942/gfx950 的测量。若未来 Inductor 对某类 stride 退化出拷贝, 这些布局会永久失去 AITER 又拿不到更快路径。
4. 判定面比目标场景更宽: `flatten_to_2d_is_free` 对所有 `rank > 2` 的输入生效, 不只 per-head QK norm。作者用 `dispatch` 探针证明 gemma-4 的连续 rank-3 视觉塔激活 `[1, 2048, 1152]` 仍被接受、只有 `strided slice` 被拒, 但任何新出现的非平凡 stride 布局都会静默改走 native, 需要靠基准而非静态阅读发现问题。
5. 缺少自动化 `dispatch` 断言: 新增测试只验证谓词与 `reshape` 是否保留 `data_ptr` (8 个 case), 端到端「哪个 provider 被选中」完全依赖作者临时插桩的探针, 仓库没有回归测试锁住 `dispatch` 决策; 若日后有人调整优先级列表或谓词顺序, 现有测试不会报警。
6. 跨平台无实测: CUDA 与 RDNA4 的不受影响结论来自结构性推理 (`AITER_SUPPORTED`、`get_default_ir_op_priority`), 作者明确表示没有 NVIDIA 硬件

；推理链条可信但属于未测量的假设。 - 影响：用户侧：ROCm (MI325X/MI355X 等 gfx942/gfx950) 上启用 VLLM_ROCM_USE_AITER=1 的用户默认命中 (VLLM_ROCM_USE_AITER_RMSNORM 默认 True)。Qwen3-0.6B/8B/14B-FP8、Qwen3-30B-A3B、llm2-1.2b、gemma-4 系列端到端吞吐提升约 +2.4%~+6.9%，每模型 dispatch 探针显示 2/3~4/6 个布局被拒；MLA 系 (DeepSeek-V4-Flash、MiniMax-M2.7、GLM-5.2、Kimi-Linear) 与无 QK norm 的模型 (Mistral-7B) 为实测 no-op。系统侧：改动局限于 ROCm AITER 的 IR op 支持谓词，不改变算子数学定义、不碰 CUDA/C++ 与共享 dispatch 代码，两个变体共享同一份编译产物；数值输出有微小差异但评测证据显示落在采样噪声内。团队侧：作者在评论中建立了一套可复用的评测范式 (dispatch 探针 + 交替顺序的 base/patched 配对基准 + 独立 VLLM_CACHE_ROOT / TORCHINDUCTOR_CACHE_DIR + gsm8k 双机回归)，这些方法细节对后续 ROCm 内核路由类 PR 有直接参考价值；同时暴露出 vllm_c 兜底路径的同类拷贝缺陷，可沉淀为独立 issue/PR。兼容性：谓词收紧只会让 AITER 在特定布局上不被选中，回退目标 native 为默认且被验证过，不构成 API 或权重兼容性风险。 - 风险标记：vllm_c 兜底仍拷贝，数值精度依赖 Inductor 生成代码，承重假设未经跨版本验证，缺少自动化 dispatch 断言，CUDA/RDNA4 仅推理未实测

关联脉络

- PR #53940 [ROCm][Kimi-K3] Enable a4w4 flydsl kernels for KimiK3: 同样在 vllm/_aiter_ops.py 与 MoE/ 量化 oracle 中裁决 AITER 内核何时被选中，属于 AITER 内核路由与开关这条功能线的直接前序工作；本 PR 补充的正是该裁决层缺失的布局收益性条件。
- PR #55358 [Refactor][GLM-5.3-Flash] Move sparse_attn_indexer_kpool into the model folder and split AMD/NVIDIA: 同属「平台特定内核按 AMD/NVIDIA 拆分与路由」的演进脉络：该 PR 把索引器拆成 amd/nvidia/common 三份实现，本 PR 则在 dispatch 谓词层区分 ROCm AITER 与其他后端。
- PR #57112 [ROCm][CI] Fix MLA RoPE fused-kernel tests for TheRock image: 同一时间窗内的 ROCm 融合内核测试工作，都涉及 ROCm 内核在不同实现 / 镜像下的验证方式；该 PR 调整 ULP 分级容差，本 PR 补充 dispatch 层测试与精度回归，主题相邻。
- PR #56904 [Bugfix] Make GPU sync checks safe under torch.compile: 与 torch.compile 全图下的运行时行为相关，本 PR 的收益同样依赖 VllmIRLoweringPass 的降级路径与 Inductor 对任意 stride 的处理，两者的风险面都落在编译期决策改变运行时内核选择。