

PR #45000 完整报告

vllm-project/vllm

[Perf][ROCm] Fix GDN KKT warmup regression on RDNA by avoiding fp32 tl.dot

合并时间: 2026-07-14 11:48

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/45000>

执行摘要

- 一句话: 修复 RDNA GPU 上 GDN KKT 内核编译时间爆炸问题
- 推荐动作: 值得精读, 尤其是对 Triton 内核性能调优感兴趣的读者。本 PR 展示了如何规避 Triton-on-ROCm 的编译降级问题, 并展示了条件编译和平台检测的最佳实践。同时, 也反映了当工具链 (Triton) 存在 bug 时, 如何通过快速、精准的工作区恢复性能。

功能与动机

PR #42076 之后, Qwen3.5-9B 模型在 RDNA GPU 上的 GDN Triton 内核预热时间从约 150 秒增加到约 2400 秒。问题在于 `tl.dot()` 的操作数被转换为 fp32, 导致该操作降级为软件 fp32 矩阵乘而非 WMMA 指令。本 PR 仅针对 RDNA GPU 修改内核, 其他 GPU 不受影响。

实现拆解

1. 平台检测与条件编译: 在文件 `vllm/model_executor/layers/fla/ops/chunk_scaled_dot_kkt.py` 中, 新增模块级变量 `_CAST_DOT_TO_K_DTYPE`, 初始为 `False`。通过 `current_platform.is_rocm()` 和 `on_gfx1x()` 判断当前 GPU 是否为 RDNA 架构 (`gfx11xx/gfx12xx`), 仅在 RDNA 上将变量设为 `True`。
2. 内核扩展: 为 `chunk_scaled_dot_kkt_fwd_kernel` 添加 `CAST_DOT_TO_K_DTYPE`: `tl.constexpr` 编译时常量参数。
3. 内核内条件逻辑: 在 `tl.dot` 调用处, 根据 `CAST_DOT_TO_K_DTYPE` 分支:
 - RDNA 路径: 将 `b_kb` 转换为 `b_k.dtype`, 即模型的本地 dtype (`bf16/fp16`), 使得 `tl.dot` 的两个操作数均为 16 位, 从而触发 WMMA 指令。
 - 其他路径: 保持原有逻辑, 将 `b_k` 转换为 `b_kb.dtype` (`fp32`), 使用 WGMMMA/MFMA 指令。
4. 调用处传递常量: 在 `chunk_scaled_dot_kkt_fwd` 函数中, 将 `CAST_DOT_TO_K_DTYPE=_CAST_DOT_TO_K_DTYPE` 传入内核启动参数。

关键文件:

- `vllm/model_executor/layers/fla/ops/chunk_scaled_dot_kkt.py` (模块 模型执行器; 类别 `source`; 类型 `core-logic`; 符号 `chunk_scaled_dot_kkt_fwd_kernel`, `chunk_scaled_dot_kkt_fwd`, `_CAST_DOT_TO_K_DTYPE`): 核心变更文件。修改了 GDN KKT Triton 内核, 通过条件编译避免 RDNA GPU 上 fp32 `tl.dot` 导致的编译时间爆炸。

关键符号: chunk_scaled_dot_kkt_fwd_kernel, chunk_scaled_dot_kkt_fwd, on_gfx1x

关键源码片段

[vllm/model_executor/layers/fla/ops/chunk_scaled_dot_kkt.py](#)

核心变更文件。修改了 GDN KKT Triton 内核，通过条件编译避免 RDNA GPU 上 fp32 tl.dot 导致的编译时间爆炸。

```
# vllm/model_executor/layers/fla/ops/chunk_scaled_dot_kkt.py

import torch
from vllm.platforms import current_platform
from vllm.triton_utils import tl, triton
from .index import prepare_chunk_indices
from .op import exp
from .utils import FLA_CHUNK_SIZE

# On RDNA (gfx11xx/gfx12xx) WMMA only
# accepts 16-bit/int inputs, so a widened (e.g. fp32) tl.dot is lowered to a
# software matmul (~190x amdgcN-stage blowup). There we cast both operands down
# to k's native storage dtype (bf16/fp16) so fast WMMA is used instead.
_CAST_DOT_TO_K_DTYPE = False
if current_platform.is_rocm():
    from vllm.platforms.rocm import on_gfx1x
    _CAST_DOT_TO_K_DTYPE = on_gfx1x()

@triton.heuristics(
    {
        # ...
    }
)
@triton.jit
def chunk_scaled_dot_kkt_fwd_kernel(
    # ...
    BK: tl.constexpr,
    IS_VARLEN: tl.constexpr,
    USE_G: tl.constexpr,
    CAST_DOT_TO_K_DTYPE: tl.constexpr,
):
    i_t, i_bh = tl.program_id(0), tl.program_id(1)
    i_b, i_h = i_bh // H, i_bh % H
    # ...
    b_k = tl.load(p_k, boundary_check=(0, 1))
    b_kb = b_k * b_beta[:, None]
    if CAST_DOT_TO_K_DTYPE:
        # RDNA: force operands to k's native dtype so WMMA is used.
        b_A += tl.dot(b_kb.to(b_k.dtype), tl.trans(b_k))
    else:
        # Keep the promoted precision of the beta-scaled operand (WGMMA/MFMA).
```

```

        b_A += tl.dot(b_kb, tl.trans(b_k).to(b_kb.dtype))
    # ...

def chunk_scaled_dot_kkt_fwd(
    # ...
):
    # ...
    chunk_scaled_dot_kkt_fwd_kernel[
        grid,
        # ...
    ](
        # ...
        CAST_DOT_TO_K_DTYPE=_CAST_DOT_TO_K_DTYPE,
    )
    return A

```

评论区精华

主要讨论集中在准确性担忧和问题根因：

- 准确性争议：审核者 ZJY0516 指出此修改可能引入精度问题，并认为本质上是 Triton-on-ROCM 的 bug。作者 nemanjaudovic 回应，已在 gfx1201 和 gfx1100 上使用 gsm8k 基准进行测试，准确率大致相同（约 0.86-0.87），并解释了 WMMA 在 RDNA 上仍以 fp32 累积，因此精度应无显著变化。
- 编译时间分析：作者测量发现，llir->amdgcN 阶段的编译时间增加了约 190 倍，这是预热时间飙升的直接原因。
- 工作区性质：评审者 dllehr-amd 认可此变更为临时工作区，期望 Triton 上游能正确支持 RDNA 场景。
- 用户反馈：外部用户确认该 PR 确实解决了 Qwen 模型在 RDNA GPU 上加载时间过长的问题。
- 准确性担忧 (correctness): 通过基准测试验证精度不变，决策为接受该修改。
- 编译时间根因分析 (performance): 当前工作区有效，上游修复待后续。

风险与影响

- 风险：回归风险：该修改仅针对 RDNA GPU，NVIDIA 和 CDNA GPU 路径完全不受影响。在 RDNA 上，由于操作数从 fp32 降级为 16 位，可能导致极小的精度差异，但作者通过 gsm8k 基准验证了准确性保持不变。性能风险：无，该修改旨在修复性能回归。兼容性风险：低，修改仅影响单个 Triton 内核。
- 影响：影响范围：仅影响使用 RDNA GPU (gfx11xx/gfx12xx) 且模型包含 GDN KKT 内核的用户（如 Qwen3.5-9B）。影响程度：显著，预热时间从 ~2400 秒恢复至 ~150 秒，提升 16 倍。正确性：通过基准测试验证，逻辑正确。
- 风险标记：核心路径变更，硬件特化修改

关联脉络

- PR #42076 引入 GDN KKT 内核的原始 PR 导致回归：本 PR 修复了由 PR #42076 引入的回归问题，故直接相关。
- PR #45929 用户报告 `make_amdgcn hang` 问题：外部用户确认本 PR 解决了其报告的问题。