PR #47770 完整报告

vllm-project/vllm

[ROCm][BugFix] Triton W4A16 handling for GPTQ/AutoGPTQ qzeros layout

合并时间：2026-07-16 00:55

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/47770

执行摘要

- 一句话：修复 ROCm Triton W4A16 GPTQ qzeros 布局错误
- 推荐动作：适合精读，尤其是处理不同量化库（GPTQ vs Compressed-Tensors）之间的张量布局兼容性时。设计决策体现了对多种 checkpoints 格式的防御性编程，值得参考。

功能与动机

用户在 issue #47159 中报告，使用 Qwen 3.6 27B GPTQ 模型时，vLLM 因 qzeros 形状断言失败而崩溃。原因是 Triton W4A16 内核假设 qzeros 始终为压缩张量布局 [N//8, K//group_size]，而 GPTQ 模型可能直接以内核期望布局 [K//group_size, N//8] 加载 qzeros。此外，对称量化模型（如 uint4b8）的 qzeros 参数本不应参与计算，但加载器仍可能注册该参数，导致错误使用。

实现拆解

1. 形状自适应：在 process_weights_after_loading 方法中，检查 zp.data.shape：若为 (K//G, N//8) 则保持；若为 (N//8, K//G) 则转置；否则抛出清晰断言错误。
2. 对称量化忽略 qzeros：在 apply_weights 方法中，对于有偏的对称量化类型（c.weight_type.has_bias()），将 qzeros 参数强制设为 None，确保内核使用标量 zp_bias 并运行 HAS_ZP=False 路径。
3. 回归测试：新增两个 ROCm 专用测试：test_triton_w4a16_process_weights_after_loading_keeps_gptq_qzeros_layout 验证 GPTQ 布局在处理后保持；test_triton_w4a16_symmetric_apply_ignores_qzeros 验证对称量化时 apply_weights 忽略 qzeros 参数。

关键文件：

- vllm/model_executor/kernels/linear/mixed_precision/triton_w4a16.py（模块 量化内核；类别 source；类型 data-contract；符号 process_weights_after_loading, apply_weights）：核心修改：qzeros 形状自适应与对称量化参数忽略
- tests/kernels/quantization/test_triton_w4a16.py（模块 测试；类别 test；类型 test-coverage；符号 test_triton_w4a16_process_weights_after_loading_keeps_gptq_qzeros_layout, test_triton_w4a16_symmetric_apply_ignores_qzeros, DummyLayer, fake_gemm）：新增两个回归测试，覆盖 GPTQ 布局保持和对称量化忽略 qzeros

关键符号：process_weights_after_loading, apply_weights, test_triton_w4a16_process_weights_after_loading_keeps_gptq_qzeros_layout, test_triton_w4a16_symmetric_apply_ignores_qzeros

关键源码片段

vllm/model_executor/kernels/linear/mixed_precision/triton_w4a16.py

核心修改：qzeros 形状自适应与对称量化参数忽略

```
# vllm/model_executor/kernels/linear/mixed_precision/triton_w4a16.py
# 在 process_weights_after_loading 方法内：

if self.w_zp_name is not None:
    zp = getattr(layer, self.w_zp_name, None)
    if zp is not None:
        c = self.config
        K, N = c.partition_weight_shape
        group_size = c.group_size if c.group_size != -1 else K
        # 内核期望的 qzeros 形状：[K // group_size, N // 8]
        expected_shape = (K // group_size, N // 8)
        # Compressed-Tensors 格式存储的转置形状：[N // 8, K // group_size]
        transposed_shape = (N // 8, K // group_size)

        if tuple(zp.data.shape) == expected_shape:
            # GPTQ/AutoGPTQ 已直接存储内核所需布局，无需转置
            qzeros = zp.data.contiguous()
        elif tuple(zp.data.shape) == transposed_shape:
            # Compressed-Tensors 存储的是转置版本，需要转置回来
            qzeros = zp.data.t().contiguous()
        else:
            raise AssertionError(
                f"{self.w_zp_name} shape mismatch: {zp.data.shape}; "
                f"expected {expected_shape} or {transposed_shape}"
            )

        replace_parameter(
            layer,
            self.w_zp_name,
            torch.nn.Parameter(qzeros, requires_grad=False),
        )
```

评论区精华

review 中 hongxiayang 询问变更是否 ROCm 专用（行 444），hnhyzz 回复“It is not rocm specific. CUDA may also get this error.”，确认问题不限于 ROCm，但 PR 测试仅针对 ROCm 启用，CUDA 路径未覆盖。

- 变更是否 ROCm 专用 (question): 确认问题不限于 ROCm，但当前测试仅覆盖 ROCm。

风险与影响

- 风险：主要风险是兼容性：若未来新的量化格式使用不同的 qzeros 布局，当前的两分支判断可能不足以覆盖，但 assertion 会提供清晰错误信息。此外，测试仅针对 ROCm 平台，CUDA 上相同问题可能仍存在（虽然逻辑相同但未测试）。性能风险极低，仅增加一次形状比较和条件分支。
- 影响：直接影响：修复了 ROCm 上 GPTQ/AutoGPTQ 量化模型的加载错误，使此类模型在 ROCm 上可用。间接影响：对压缩张量格式（Compressed-Tensors）的行为不变，因为转置逻辑仍兼容。对对称量化模型，行为被修正，不再错误使用 qzeros 参数。用户无需修改配置，升级即可生效。
- 风险标记：仅 ROCm 测试，兼容性风险低

关联脉络

- 暂无明显关联 PR