

PR #47116 完整报告

vllm-project/vllm

[XPU][Bugfix] Do not transpose weight_scale_inv at load time

合并时间: 2026-07-06 17:15

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/47116>

执行摘要

- 一句话: 移除 `weight_scale_inv` 加载时转置, 修复 DeepSeek 精度
- 推荐动作: 该 PR 是典型的数据布局契约修复: 检查点格式与 kernel 期望格式不匹配, 且加载时转换方式错误。值得精读以理解 FP8 块级量化的布局约定及典型 bug 模式。建议未来增加单元测试验证 `scale` 布局的正确性, 并确认 `e8m0→float32` 转换的移除对其他模型的影响。

功能与动机

DeepSeek-V4-Flash 等 MLA 模型的 `_o_proj` 路径通过 `_get_cached_wo_a_bf16` 直接读取 `weight_scale_inv`, 并假定其为 checkpoint 原生布局 `[N/128, K/128]` 以进行手动反量化。加载时转置为 `[K/128, N/128]` 后, `_expand_2d_block_scales` 使用了错误的 block 位置, 导致精度严重下降 (GSM8K: 96.5% → 89%)。PR Body 提供了详细根因分析和示例。

实现拆解

1. 移除 `process_weights_after_loading` 方法 (`xpu.py`, 删除 ~24 行): 该方法在 `XPUFp8BlockScaledMMKernel` 中被覆盖, 其中调用 `replace_parameter(layer, scale_attr, scale.data.t().contiguous())` 将 `weight_scale_inv` 进行了转置并原地替换。PR 直接移除了整个方法覆盖, 从而保留 checkpoint 的原始 `[N/128, K/128]` 布局。
2. 在 `apply_block_scaled_mm` 中添加运行时转置 (`xpu.py`, 新增 2 行): 在 `oneDNN fp8_gemm` 调用之前, 将 `Bs` (即 `weight_scale_inv`) 通过 `Bs.t().contiguous()` 转置为 `oneDNN` 期望的 `[K/128, N/128]` 布局。由于 `scale` 张量通常很小 (约 3.5KB), 转置开销可忽略。
3. 保留 `e8m0` 转 `float32` 转换 (但已移除其所在的方法): 原 `process_weights_after_loading` 中还包含将 `float8_e8m0fnu` 转换为 `float32` 以触发 `block-quant` 路径的逻辑。这一转换在方法移除后不再生效。然而, 根据 PR 讨论和代码上下文, 关键模型 (如 DeepSeek-V4) 不再需要此转换, 因为 `apply_block_scaled_mm` 中的运行时转置已解决核心问题。若后续出现 `e8m0` 精度模型, 需单独处理。
4. 测试配套: 本次变更未增加新的测试覆盖。风险在于缺乏针对运行时转置正确性的验证。

关键文件:

- `vllm/model_executor/kernels/linear/scaled_mm/xpu.py` (模块 量化内核; 类别 `source`; 类型 `data-contract`; 符号 `process_weights_after_loading`, `apply_block_scaled_mm`):

唯一变更文件：核心修复在此文件中实现。移除了 `process_weights_after_loading` 方法，并在 `apply_block_scaled_mm` 中添加了运行时转置。

关键符号：XPUFp8BlockScaledMMKernel.process_weights_after_loading,
XPUFp8BlockScaledMMKernel.apply_block_scaled_mm

关键源码片段

vllm/model_executor/kernels/linear/scaled_mm/xpu.py

唯一变更文件：核心修复在此文件中实现。移除了 `process_weights_after_loading` 方法，并在 `apply_block_scaled_mm` 中添加了运行时转置。

```
# vllm/model_executor/kernels/linear/scaled_mm/xpu.py
class XPUFp8BlockScaledMMKernel(Fp8BlockScaledMMLinearKernel):
    # ... is_supported 保持不变 ...

    # 移除了 process_weights_after_loading 方法覆盖,
    # 因此不再在加载时对 weight_scale_inv 进行转置。
    # 现在 weight_scale_inv 保持 checkpoint 原生布局
    # [N/128, K/128] (N 为输出维度, K 为输入维度) 。

    def apply_block_scaled_mm(
        self,
        A: torch.Tensor,
        B: torch.Tensor,
        As: torch.Tensor,
        Bs: torch.Tensor,
    ) -> torch.Tensor:
        # Weight is [N, K]. Use .t() to create a [K, N] view without copying.
        # Bs ( 即 weight_scale_inv) 在加载后保持原生布局 [N/128, K/128]。
        # 但在传递给 oneDNN fp8_gemm 之前, 需要转置为内核期望的
        # [K/128, N/128] 布局, 以确保 block 映射正确。
        # 转置开销很小 (约 3.5KB) 。
        return torch.ops._xpu_C.fp8_gemm(
            A,
            B.t(),
            self.config.out_dtype,
            As,
            Bs.t().contiguous(), # 运行时转置, 而非加载时
            torch.Tensor(),
        )
```

评论区精华

PR 的 review 评论较少，主要由审核者 [xinyu-intel](#)、[zufangzhu](#) 和 [jikunshang](#) 批准。
Jikunshang 在 Issue 评论中提及本 PR 的合入有助于节省 CI 资源。无实质性争议或设计讨论。

- 暂无高价值评论线程

风险与影响

- 风险:

1. 回归风险: 移除 `process_weights_after_loading` 会同时删除 `e8m0→float32` 的转换逻辑。若其他模型（非 DeepSeek-V4）依赖该转换来触发 `block-quant` 路径，则可能退化为 `per-channel` 反量化，导致 NaN 或精度损失。本 PR 的分析者假设当前所有 XPU MLA 模型均使用 `float32 weight_scale`，但缺乏确认。
2. 运行时转置开销: `Bs.t().contiguous()` 会引入一次小的内存拷贝，但由于 `scale` 张量很小，通常可以忽略。
3. 缺少测试覆盖: 没有针对 `apply_block_scaled_mm` 中尺度布局正确性的自动化测试，回归依赖手动验证。
 - 影响: 直接影响 Intel GPU (XPU) 上使用 `XPUFp8BlockScaledMMKernel` 的 MLA 模型（如 DeepSeek-V4-Flash）。修复了精度严重下降的回归 bug，对用户而言是正收益。影响范围集中于 MLA attention 路径，其他模型和量化方案不受影响。系统层面，无外部接口变更。
 - 风险标记: 核心路径变更，缺少测试覆盖

关联脉络

- 暂无明显关联 PR