

PR #56433 完整报告

vllm-project/vllm

[ROCm][Bugfix] Fix AITER preshuffled FP8 block-scale kernel

合并时间: 2026-09-11 19:44

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/56433>

执行摘要

- 一句话: 修复 ROCm AITER 预重排 FP8 内核错误, 恢复 MLA 模型精度
- 推荐动作: 建议精读, 尤其是维护 ROCm、量化 GEMM 或 MLA 路径的工程师: 这是典型的「内核与模型各做一次权重变换」类静默 bug, 修复手法 (能力标志 + 运行时兜底 + 生产者 - 消费者布局协商 + 拒绝不一致快路径) 可直接迁移到其他平台后端。重点看三处设计决策: 一是为什么 `is_bmm/skip_weight_relayout` 必须在 `process_weights_after_loading` 而不是 `can_implement` 里判断; 二是 `_wq_b_act_scale_transpose` 返回 `bool | None`、把「无法协商」显式编码为回退信号; 三是 `apply_block_scaled_mm` 的 `byte-order` 契约只有 `docstring` 兜底。后续行动建议: 补一个 `TP >= 4` 的 DeepSeek/Kimi 端到端或 kernel 级回归测试 (校验行 / 列主序 `As` 的数值差异), 并确认 `_reads_weight_directly` 提前 `return` 时跳过 `e8m0 upcast` 是否有意为之。

功能与动机

PR #51692 引入 `AiterPreshuffledFp8BlockScaledMMKernel` 并把它放在 ROCm `block-scaled` 优先级列表首位, 导致 DeepSeek-V4/V3、Kimi-K2 等大量 MLA 模型自动选中该内核, 从而暴露四处缺陷 (PR body 原文): 内核「reads the activation scale as column-major, while pre-quantizing callers emit row-major scales」, 两者形状恰好相同 (body 写作 `(M, N // 32)`) 所以不会报错; 内核在加载后 B-preshuffle 权重, 而模型又对 `fused_wqa_wkv`、`wo_b`、`gate_up_proj` 二次 shuffle, 而 `shuffle_weight` 是置换而非对合; `wo_a` 是 BMM 层、权重被模型直接读取, 被预重排后输出投影损坏; `kv_b_proj` 被 MLA 经 `get_and_maybe_dequant_weights` 读取以构造 `W_UK/W_UV`, 同样不能被重排。此外 DeepSeek-V4 在 ROCm 上因 `_wq_b_proj` 命中 `apply_block_scaled_mm` 抛 `NotImplementedError` 而完全无法启动。症状与 TP 规模相关: DS V3/R1 在 TP1/TP2 正常、TP4/TP8 出错, Kimi K2 在 TP1 正常、TP>2 出错, 因此原先的 DS V3 TP1/DP8 测试没有覆盖到腐蚀区间。

实现拆解

1. 内核侧把隐式约定升级为显式能力标志 (`vllm/model_executor/kernels/linear/scaled_mm/aiter.py`): 在 `AiterPreshuffledFp8BlockScaledMMKernel` 上新增 `wants_transposed_act_scale: ClassVar[bool] = True` (该 GEMM 按列主序读激活 `scale`) 与 `preshuffles_weight: ClassVar[bool] = True` (加载期会把 `layer.weight` 重排成 (16, 16)), 并新增静态判定 `_reads_weight_directly(layer)`, 通过 `is_bmm` (`wo_a` 权重栈) 与

`skip_weight_relayout` (MLA 的 `kv_b_proj`) 识别「权重会被 `apply_weights` 之外的代码直接读取」的层。这两个标志都是在层构造之后才打上的, 所以 `can_implement` 看不到, 只能在加载期用运行时判断兜底。

2. 加载期与执行期按权重布局分叉: `process_weights_after_loading` 对 `_reads_weight_directly` 为真的层提前 return, 不再做 `shuffle_weight`; `apply_weights` 中新增 `plain = self._reads_weight_directly(layer)`, `plain` 路径用 `rocm_aiter_ops.gemm_a8w8_blockscale` (激活量化 `transpose_scale=False`), 普通路径仍用 `gemm_a8w8_blockscale_bpreshuffle` (`transpose_scale=True`), 保证「权重布局」与「激活 `scale` 字节序」始终成对一致。原先 `apply_block_scaled_mm` 直接 raise `NotImplementedError`, 现在改为调用 `gemm_a8w8_blockscale_bpreshuffle`, 并在 docstring 明确警告 As 必须是列主序、行主序不会报错但 $M > 1$ 时返回错误数值——这条接口正是 DeepSeek-V4 `_wq_b_proj` 预量化路径的入口。
3. 模型侧跳过重复 shuffle: `vllm/models/deepseek_v4/amd/rocm.py` 新增模块级函数 `weight_already_preshuffled(linear)` (读 `kernel.preshuffles_weight`), `prepare_attn_preshuffle._prep` 在处理 `fused_wqa_wkv`、`wo_b` 时先判断再 `replace_parameter`; `vllm/models/deepseek_v4/amd/model.py` 的 `DeepseekV4MLP.prepare_gateup_preshuffle` 复用同一判断保护 `gate_up_proj`。这样内核与模型两侧谁做 shuffle 由内核能力标志决定, 而不是靠模型名硬编码。
4. 生产者 - 消费者激活 `scale` 主序协商: `rocm.py` 把 `_wq_b_uses_aiter_block_scaled` (bool) 替换为 `_wq_b_act_scale_transpose` (bool | None)。它收集 `self.wq_b` 与 `indexer.wq_b` 两个消费者的 `wants_transposed_act_scale`; 若两者主序不一致就 `logger.warning_once` 并返回 None, 主动放弃融合快路径——因为一份 `qr_scale` 无法同时满足两个内核。`_split_qkv_and_norm` 据此把传给 `rocm_aiter_ops.fused_qk_rmsnorm_group_quant` 的 `transpose_scale` 从硬编码 `False` 改为协商结果, 并在 `transpose_scale is None` 时回退共享路径。
5. 跨后端标志改名: `vllm/model_executor/layers/attention/mla_attention.py` 把 `kv_b_proj._cpu_skip_gemm_dispatch = True` 改为 `kv_b_proj.skip_weight_relayout = True`, 注释也从「AMX/CPU 语义」改为通用理由「MLA reads this weight directly to build `W_UK/W_UV`, so a backend must not relayout it at load time」; `vllm/model_executor/layers/linear.py` 的 CPU `process_weights_after_loading` 随之改读新标志, 跳过打包、退化为 `torch.nn.functional.linear`。
6. 测试与验证配套: 本 PR 不含任何测试文件或配置改动 (`changed_files` 全为源码), 验证完全依赖手工结果与 CI: PR body 给出 DeepSeek V4 / V3 TP8 gsm8k 由 0 恢复到 0.9568 / 0.9458, DS V4 另由 AMD CI 的「LM Eval (Large Models)」测试组覆盖。

关键文件:

- `vllm/model_executor/kernels/linear/scaled_mm/aiter.py` (模块 线性内核; 类别 source; 类型 core-logic; 符号 `AiterPreshuffledFp8BlockScaledMMKernel`, `wants_transposed_act_scale`, `preshuffles_weight`, `_reads_weight_directly`): 内核主改动文件: 新增 `preshuffles_weight` / `wants_transposed_act_scale` 两个能力标志与 `_reads_weight_directly` 运行时判定, 让 `bpreshuffle` 内核对被直接读取的权重保持原布局, 并把 `apply_block_scaled_mm` 从抛 `NotImplementedError` 改为可用的 `bpreshuffle`

GEMM，直接决定 DeepSeek-V4 能否在 ROCm 启动。

- `vllm/models/deepseek_v4/amd/rocm.py` (模块 注意力实现; 类别 source; 类型 data-contract; 符号 `weight_already_preshuffled`, `prepare_attn_preshuffle`, `_wq_b_act_scale_transpose`, `_split_qkv_and_norm`) : 模型侧主改动: 新增 `weight_already_preshuffled` 供 `prepare_attn_preshuffle` 跳过二次 shuffle, 并把 `_wq_b_uses_aiter_block_scaled` 升级为 `_wq_b_act_scale_transpose`, 在两个 `wq_b` 消费者主序不一致时主动放弃融合 `norm+quant` 快路径——这是消除「激活 scale 主序不匹配」这类静默错误的核心逻辑。
- `vllm/models/deepseek_v4/amd/model.py` (模块 前馈网络; 类别 source; 类型 core-logic; 符号 `DeepseekV4MLP.prepare_gateup_preshuffle`) : `DeepseekV4MLP.prepare_gateup_preshuffle` 复用 `weight_already_preshuffled`, 避免 `gate_up_proj` 权重被内核与模型重复 shuffle; 同时把该判断导入模型文件, 是「模型按内核能力决定布局」模式的另一处落地。
- `vllm/model_executor/layers/attention/mla_attention.py` (模块 注意力层; 类别 source; 类型 data-contract; 符号 `MLAAttention.init`) : 把 `kv_b_proj._cpu_skip_gemm_dispatch = True` 改为后端无关的 `kv_b_proj.skip_weight_relayout = True`, 注释从 AMX/CPU 专属理由改为「MLA 直接读取该权重构造 `W_UK/W_UV`」, 是内核侧 `_reads_weight_directly` 能识别 `kv_b_proj` 的前提。
- `vllm/model_executor/layers/linear.py` (模块 线性层; 类别 source; 类型 data-contract; 符号 `UnquantizedLinearMethod.process_weights_after_loading`) : `UnquantizedLinearMethod.process_weights_after_loading` 的 CPU 分支同步改读 `skip_weight_relayout`, 保证标志改名后 CPU 上 MLA 的 `kv_b_proj` 仍跳过打包、退化为普通 `F.linear`。

关键符号: `weight_already_preshuffled`, `AiterPreshuffledFp8BlockScaledMMKernel._reads_weight_directly`, `AiterPreshuffledFp8BlockScaledMMKernel.process_weights_after_loading`, `AiterPreshuffledFp8BlockScaledMMKernel.apply_weights`, `AiterPreshuffledFp8BlockScaledMMKernel.apply_block_scaled_mm`, `DeepseekV4ROCMaIterMLAAttention._wq_b_act_scale_transpose`, `DeepseekV4ROCMaIterMLAAttention._split_qkv_and_norm`, `DeepseekV4ROCMaIterMLAAttention.prepare_attn_preshuffle`, `DeepseekV4MLP.prepare_gateup_preshuffle`, `UnquantizedLinearMethod.process_weights_after_loading`

关键源码片段

`vllm/model_executor/layers/attention/mla_attention.py`

把 `kv_b_proj._cpu_skip_gemm_dispatch = True` 改为后端无关的 `kv_b_proj.skip_weight_relayout = True`, 注释从 AMX/CPU 专属理由改为「MLA 直接读取该权重构造 `W_UK/W_UV`」, 是内核侧 `_reads_weight_directly` 能识别 `kv_b_proj` 的前提。

```
self.q_pad_num_heads = getattr(self.impl, "q_pad_num_heads", None)
self.is_amx_bmm_enabled = getattr(self.impl, "uses_amx_bmm", False)
```

```
# 旧实现只表达「CPU/AMX 不要打包」的私有意图 (_cpu_skip_gemm_dispatch) ;
# 现在改为后端无关的 skip_weight_relayout: MLA 会直接读取该权重来
# 构造 W_UK / W_UV, 所以任何后端都不得在加载期对它做重排。
# 该属性随后由 AiterPreshuffledFp8BlockScaledMMKernel._reads_weight_directly
# 与 UnquantizedLinearMethod.process_weights_after_loading 共同消费。
kv_b_proj.skip_weight_relayout = True
self.use_direct_call = not current_platform.opaque_attention_op()
```

评论区精华

本 PR 的讨论量很小: 3 条 issue 评论全部是流程性内容 (tjtanaa 的 `/ci run` 与两条 CI 机器人回执), `review_comments` 为 0, 没有任何行级讨论。两条 review 分别是 `claude[bot]` 的 COMMENTED (原话: "This pull request is from a fork — automated review is disabled.") 和 reviewer tjtanaa 的 APPROVED, 批注只有一句: "Approved as hot fix."。因此能力标志的时序问题 (`is_bmm/skip_weight_relayout` 在构造后才打标、`can_implement` 看不到)、`apply_block_scaled_mm` 的「行主序 As 不报错但算错」契约、以及缺失回归测试这三点都没有被评审讨论触及, 属于「代码本身写清了理由、但没有人反驳或追问」的状态。

- 以 hotfix 身份快速批准合入 (other): 作为阻塞 ROCm MLA 模型的热修被接受并合入 (`merged_by` 同为 tjtanaa), 但能力标志的时序问题、`apply_block_scaled_mm` 的静默 byte-order 契约等设计细节没有经过对撞式评审。
- fork PR 的自动评审被禁用 (other): 未触发一次性自动评审, 本 PR 的评审覆盖完全依赖人工批准与 CI (tjtanaa 的 `/ci run` 与 Buildkite 回执)。
- 缺失回归测试未被讨论 (testing): 测试缺口保持未解决: 该缺陷只在 $M > 1 / TP > 2$ 时显现, 恰恰是最需要自动化覆盖的区间, 建议后续补 $TP \geq 4$ 的端到端或 kernel 级主序回归测试。

风险与影响

- 风险: 1) 缺少自动化回归测试: PR 修改了 5 个文件却没有任何测试变更, 而这类缺陷只有在 $M > 1$ 或 $TP > 2$ 时才显现 (DS V3/R1 在 TP4/TP8 出错、Kimi K2 在 $TP > 2$ 出错), 恰好是单测最容易漏掉的区间; 当前只有 DS V4/V3 TP8 的手工 gsm8k 结果, 未来任何按 TP 形状切换内核 (`is_blockscale_bpreshuffle_tuned(n, k)`) 的改动都可能重新引入静默错误。2) 静默接口契约: `AiterPreshuffledFp8BlockScaledMMKernel.apply_block_scaled_mm` 的 docstring 明确写了「row-major one will not raise, it just returns wrong numbers for $M > 1$ 」, 调用方 `apply_pre_quantized_block_scaled_mm` 必须依赖 `_wq_b_act_scale_transpose` 才能给出正确的 As, 任何新增调用者若忽略该约定不会得到报错。3) 提前 return 跳过了权重 scale 上采样: `process_weights_after_loading` 在 `_reads_weight_directly` 为真时于 `ws.dtype == torch.float8_e8m0fnu` 的 upcast 之前 return, 而这些层之后仍会走 `apply_weights` 的 plain 分支直接调用 `gemm_a8w8_blockscale`, 不像 `Fp8BlockScaledMMLinearKernel.apply_block_scaled_mm` 那样统一 dtype; 若这些层使用 e8m0 权重 scale, 可能存在未覆盖的 dtype 隐患 (材料未提供该场景的验证, 属于需确认项)。4) 标志改名的兼容性: `_cpu_skip_gemm_dispatch` $\rightarrow$ `skip_weight_relayout` 是私有属性重命名, 若仓库其他位置或外部插件仍设置旧名, CPU 路径会静默退回打包分支 (行为与性能变化), 需要 grep 确

认已全部迁移。5) 性能回退：当 wq_b 与 indexer.wq_b 主序不一致时融合 norm+quant 快路径被禁用（每位进程只警告一次），退回共享路径会额外读取并量化 bf16 qr; kv_b_proj 等层也退回普通 blockscale GEMM, prefill-context 路径可能略慢，属于正确性优先的取舍。6) ROCm 专属面：改动只影响 ROCm + AITER + fp8 block-scaled 组合，CUDA/CPU 路径不受影响，回归风险被限制在 AMD 平台，但也意味着非 AMD CI 基本不会发现该文件的回归。

- 影响：用户侧：ROCm 上运行 DeepSeek V3/R1、V4、Kimi-K2 等 MLA + fp8 block-scaled 模型、并使用 TP4/TP8 的用户此前拿到的是完全不可用的输出（gsm8k 0，且不报错），DeepSeek-V4 在 ROCm 上则直接无法启动；修复后精度恢复到 ~0.95，属于「从不可用恢复到可用」的量级。系统侧：把「权重是否已被内核重排」「激活 scale 用哪种主序」从散落在模型代码里的隐式假设，提升为内核能力标志（preshuffles_weight、wants_transposed_act_scale）与层属性（skip_weight_relaxation），并把 MLA 加载期契约从 CPU/AMX 专属语义改为后端通用语义，这类契约变更会被随后新增的 kernel/backend 继承。团队侧：aiter.py、mla_attention.py、linear.py 的改动建立了「内核声明能力、模型按能力决定是否手工 relax」的协作模式，后续做新 AITER 内核或新量化布局时需要遵循，否则容易重现本 PR 的静默错误；同时 AMD CI 的「LM Eval (Large Models)」组承担了事实上的回归防线。
- 风险标记：缺少测试覆盖，静默数值错误风险，跨模块契约变更，ROCm 专属路径，潜在性能回退

关联脉络

- PR #51692 引入 AiterPreshuffledFp8BlockScaledMMKernel 并置于 ROCm block-scaled 优先列表首位（标题未在材料中提供，据本 PR body 引用）：本 PR 修复的回归源头：正是它把 bpreshuffle 内核排到优先列表首位，才使 DeepSeek-V4/V3、Kimi-K2 等 MLA 模型选中该内核并暴露激活 scale 主序、二次 shuffle、wo_a/kv_b_proj 重排四处缺陷。
- PR #56214 [Model] Support DeepSeek-V4.1-Flash: 同一 DeepSeek V4 模型线的近期改动（ROCm/MXFP8 路径、MLA indexer、kv_cache_utils），与本 PR 修改的 vllm/models/deepseek_v4/amd/* 共享模型与量化内核选择逻辑，本 PR 修复的 DS V4 启动问题正属于该模型线。
- PR #55239 [ROCm][Bugfix] Route GLM-5.3-Flash MTP through ragged sparse MLA: 同为 ROCm MLA/AITER 路径上的静默数值错误修复（放宽 sparse-MLA 分派谓词），且 rocm.py 直接导入 vllm/v1/attention/ops/rocm_aiter_mla_sparse，属于同一「ROCm 内核分派选错即算错」的问题族。
- PR #56138 [Bugfix] Pin EPLB and MLA host-to-device transfer buffers: 近期 MLA / 稀疏注意力路径的 bugfix 且同时补了回归测试（tests/v1/attention、tests/distributed），可作为本 PR 缺失测试的对照：同类 ROCm/MLA 静默正确性问题通常都配了自动化回归。