

PR #51692 完整报告

vllm-project/vllm

[ROCm][Perf] Add bpres shuffled block scaled fp8 GEMM

合并时间: 2026-09-11 00:18

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/51692>

执行摘要

- 一句话: ROCm FP8 分块 GEMM 新增预洗牌内核
- 推荐动作: 值得精读, 尤其是正在维护 ROCm FP8 线性层或负责 DSv3 类模型部署的工程师。建议重点关注三个设计决策: (1) 为什么用独立内核类而不是 `_is_bpres shuffled` 标志位 (review 结论很清晰: 布局差异 + 日志可观测性); (2) 为什么用调优表而非环境变量做启用门禁 (fallback 会 fault/ 数值错误, 宁可退回旧内核); (3) `apply_block_scaled_mm` fail-fast 的边界——这是外部反馈 V4 初始化失败的直接原因, 值得评估是否改为委托父类实现或按形状再判一次。复现时请先按 PR body 设置 `VLLM_ROCM_USE_AITER_FP8BMM=0`, 并注意需自行验证 mawong-amd 报告的 V3 TP8 输出异常是否仍然存在; 如果要在生产启用, 建议先在目标机型上用 `is_blockscale_bpres shuffle_tuned` 确认调优表命中, 再对比开启前后的 GSM8k/ 长上下文精度。后续应补一个 ROCm 侧的选型单测 (覆盖 tuned/ 未 tuned、N 或 K 非 128 倍数、e8m0 scale 上采样三个分支)。

功能与动机

动机来自 ROCm 上 DeepSeek-V3 类 FP8 block-scaled 模型的解码瓶颈: PR body 的 profiling 显示, MLA decode 的 o-proj 在 main 分支走 ck-tile 内核需要 52us, 而 bpres shuffle 版本只要 21us, 作者原文称“more than 2x slower than that of the bpres shuffled version”。收益目标是 DSv3 1k/1k 场景 TP8+DPA +4~8% QPS、TP8+EP +0~4% QPS, 并用 GSM8k 验证精度不掉 (0.9477 vs nightly 0.9439)。同时 PR body 明确说明需要 `VLLM_ROCM_USE_AITER_FP8BMM=0` 才能跑通, 直到关联 issue #51957 (AITER FP8 BMM 在 data parallel attention 下 segfault) 解决, 这也是内核实现中选择 `group_fp8_quant(..., transpose_scale=True)` 的直接原因。

实现拆解

1. 新增独立内核类 (`vllm/model_executor/kernels/linear/scaled_mm/aiter.py`): 新增 `AiterPres shuffledFp8BlockScaledMMKernel`, 继承 `Fp8BlockScaledMMLinearKernel`。`is_supported` 直接委托 `AiterPres shuffledPerTokenFp8ScaledMMLinearKernel.is_supported` (ROCm + AITER 可用即可); `can_implement` 叠加多道门禁: 父类基础校验、`activation_quant_key.scale.group_shape == GroupShape(1, 128)`、`rocm_aiter_ops.is_linear_fp8_enabled()`、N 与 K 均能被 128 整除、以及 `rocm_aiter_ops.is_blockscale_bpres shuffle_tuned(n, k)`。理由是 fallback 路径可能触发 fault 或数值错误, 因此宁可的不选也不能选错。

2. 权重布局改造（同文件 `process_weights_after_loading / apply_weights`）：先复用父类的权重预处理，再把 e8m0 的 `weight_scale(_inv)` 上采样为 fp32；随后断言权重是 2D fp8 且 N/K 均为 128 的倍数，最后用 `rocm_aiter_ops.shuffle_weight(..., layout=(16, 16))` 替换 `params.weight`，并包成 `torch.nn.Parameter(..., requires_grad=False)`（与 ROCm 上 DLPack 对 Parameter 的要求一致）。前向阶段 `apply_weights` 用 `group_fp8_quant(x_2d, transpose_scale=True)` 生成 A/As，再调用 `gemm_a8w8_blockscale_bpreshuffle`，bias 手动相加后 reshape 回原形状。
3. fail-fast 契约（同文件 `apply_block_scaled_mm`）：该方法直接 `raise NotImplementedError`，把“权重已被改成洗牌布局、不能再走通用 block-scaled 入口”这一隐含契约显式化。这是对 review 中 `_is_bpreshuffled` 标志方案的替代设计，保留了对误调用的防护，但代价是任何仍按 `apply_block_scaled_mm` 分派的模型会直接初始化失败。
4. 调优表读取通用化（`vllm/_aiter_ops.py`）：`_load_gemm_tuned_configs` 由 `(q_dtype_w, csv_path)` 改为 `(csv_path, filters, key_cols)`，逐列过滤后按 `key_cols` 组集合，缺列与读表异常都退化为空集；`_check_kernel_tuned` 改为显式传 `(('q_dtype_w', q_dtype_w),)`，保持原语义。新增静态方法 `is_blockscale_bpreshuffle_tuned(n, k)`，读取 `AITER_CONFIG_GEMM_A8W8_BLOCKSCALE_BPRESHUFFLE_FILE` 并按 `get_gfx()` + `get_cu_num()` 过滤，只以 (N, K) 为键——即同一权重形状在不同 GPU 代数上可能命中不同结论。
5. 选型注册（`vllm/model_executor/kernels/linear/init.py`）：新增 import、加入 `aiter backend` 的 kernel 集合、补充 `__all__`，并插入 `_POSSIBLE_FP8_BLOCK_KERNELS[PlatformEnum.ROCM]` 列表首位（排在 `AiterFp8BlockScaledMMKernel`、`TritonFp8BlockScaledMMKernel` 之前）。这一处才是真正决定线上默认选型的改动。
6. 测试与运行配套：本次没有新增或修改任何测试文件，验证依赖 PR body 中 8xMI350 的 bench serve（并发 1-128）、lm_eval gsm8k 以及 profiling 截图；运行配置上需要 `VLLM_ROCM_USE_AITER=1`、`VLLM_ROCM_USE_AITER_LINEAR=1`，并临时设置 `VLLM_ROCM_USE_AITER_FP8BMM=0` 规避 issue #51957。

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`, `is_supported`, `can_implement`, `process_weights_after_loading`）：本 PR 的主体：新增 `AiterPreshuffledFp8BlockScaledMMKernel`，定义权重洗牌布局、调优表门禁、前向 GEMM 调用以及 `apply_block_scaled_mm` 的 fail-fast 契约。
- `vllm/_aiter_ops.py`（模块 调优配置；类别 source；类型 core-logic；符号 `_load_gemm_tuned_configs`, `_check_kernel_tuned`, `is_blockscale_bpreshuffle_tuned`）：把调优表读取函数通用化（`filters/key_cols`），并新增 `is_blockscale_bpreshuffle_tuned`——它是新内核能否被选中的唯一数据依据，也是后续更多调优表可复用的公共入口。
- `vllm/model_executor/kernels/linear/__init__.py`（模块 内核注册；类别 source；类型 configuration；符号 `_POSSIBLE_FP8_BLOCK_KERNELS`, `all`）：注册入口：把新内核加入 `aiter backend` 集合、`__all__`，并插到 ROCm block-scaled 候选列表首位——这一行决定了线上默认选型行为的变化。

关键符号: AiterPreshuffledFp8BlockScaledMMKernel.is_supported,
AiterPreshuffledFp8BlockScaledMMKernel.can_implement,
AiterPreshuffledFp8BlockScaledMMKernel.process_weights_after_loading,
AiterPreshuffledFp8BlockScaledMMKernel.apply_weights,
AiterPreshuffledFp8BlockScaledMMKernel.apply_block_scaled_mm,
_load_gemm_tuned_configs, _check_kernel_tuned, is_blockscale_bpreshuffle_tuned

关键源码片段

vllm/_aiter_ops.py

把调优表读取函数通用化 (filters/key_cols) , 并新增 `is_blockscale_bpreshuffle_tuned`——它是新内核能否被选中的唯一数据依据, 也是后续更多调优表可复用的公共入口。

```
@functools.cache
def _load_gemm_tuned_configs(
    csv_path: str,
    filters: tuple[tuple[str, object], ...],
    key_cols: tuple[str, ...] = ("N", "K", "M"),
) -> set[tuple[int, ...]]:
    """按 filters 过滤 aiter 调优表, 再抽取 key_cols 组成集合 (按调用参数缓存) 。”
    try:
        df = pd.read_csv(csv_path).drop_duplicates()
        for col, val in filters:
            if col not in df.columns:
                continue # 不同 CSV 的列集合不同, 缺列直接跳过而不是报错
            if isinstance(val, int):
                df = df[df[col].astype(int) == val]
            else:
                df = df[df[col].astype(str) == str(val)]
        return set(zip(*(df[c].astype(int) for c in key_cols)))
    except Exception:
        return set() # 读表失败时退化为「无调优配置」, 调用方自然回退到旧内核
```

```
# vllm/_aiter_ops.py: class rocm_aiter_ops 内新增的静态方法
@staticmethod
def is_blockscale_bpreshuffle_tuned(n: int, k: int) -> bool:
    """(N, K) 是否命中当前 gfx / CU 数的 blockscale bpreshuffle 调优配置。"""
    if not current_platform.is_rocm():
        return False
    import aiter.ops.gemm_op_a8w8 as aiter_gemm_a8w8_ops

    csv_path = (
        aiter_gemm_a8w8_ops.AITER_CONFIGS.AITER_CONFIG_GEMM_A8W8_BLOCKSCALE_
        BPRESHUFFLE_FILE
    )
    gfx = aiter_gemm_a8w8_ops.get_gfx()
    cu_num = aiter_gemm_a8w8_ops.get_cu_num()
```

```
# 与 per-token 路径不同：这里按 gfx + cu_num 过滤，键只取 (N, K)，
# 因此同一权重形状在不同 GPU 代数上会有不同结论
return (n, k) in _load_gemm_tuned_configs(
    csv_path, (("gfx", gfx), ("cu_num", cu_num)), key_cols=("N", "K")
)
```

评论区精华

- 调优表读取：复用还是各写一份（design，已闭环）：Rohan138 提出 “There's a `_load_gemm_tuned_configs` helper, can we use that instead?”；作者最初倾向保留两份实现（理由：统一函数 “might become overly complex easily”），Rohan138 回复 “honestly I slightly prefer the generalization for now ... we'd eventually like to stop checking them here and have AITER ... just choose the tuned configs or fall back as needed”，最终采纳通用化方案，作者也认可“将来更容易删掉这段逻辑”。
- `_is_bpreshuffled` 标志 vs 独立内核类（design，已闭环且改变了实现形态）：BadrBasowid 质疑早退路径会让 `AiterFp8BlockScaledMMKernel.apply_weights` 吃到错误权重布局，建议挪到 `can_implement`；tjtanaa 提出 “better to create a `AiterPreshuffledFp8BlockScaledMMKernel`”；BadrBasowid 补充 “Creating a separate class would also be better for logging, since it clearly prints which kernel class has been selected”。作者据此重构成独立类，并贴出日志 `Selected AiterPreshuffledFp8BlockScaledMMKernel for Fp8LinearMethod` 作为验证。
- V3/V4 正确性反馈（correctness，未闭环）：mawong-amd 在 issue 评论中报告 “I'm seeing garbage output on the former, and it refuses to initialize on the latter (it hits the not implemented `apply_block_scaled_mm`)”，并提示 main 上 #56228 改了 `sparse-attention indexer` 签名属于另一处破坏；作者回应 “yes tested on V3 TP8 (note: vanilla dense v3). Let me also test again”，提供的材料中没有后续结论。
- CodeRabbit/Ruff ISC004（style，已闭环）：CodeRabbit 指出多处 tuple 内隐式字符串拼接未加括号，作者以 `chore: wrap string literals in parens` 修复。
- CI 资源协调（other）：AndreasKaratzas 提醒 “I cancelled this build cause like the other 2 it creates a huge workload for our CI”，要求逐 PR 跑完再开下一个；作者随后确认失败用例与 nightly 一致。
 - 用 `_is_bpreshuffled` 标志位还是独立的 `Preshuffled` 内核类 (design): 作者重构成独立内核类，commit refactor: into `AiterPreshuffledFp8BlockScaledMMKernel`，并在讨论中贴出 `Selected AiterPreshuffledFp8BlockScaledMMKernel for Fp8LinearMethod` 日志与 o-proj 仍走 `preshuffled GEMM` 的截图。
 - 调优表读取是否复用 `_load_gemm_tuned_configs` (design): 采纳通用化版本：`_load_gemm_tuned_configs(csv_path, filters, key_cols)`，`_check_kernel_tuned` 改为显式传 `filters`；新增的 `is_blockscale_bpreshuffle_tuned` 复用同一函数。
 - 新内核 `apply_block_scaled_mm` 未实现导致 V4 初始化失败、V3 TP8 输出异常 (correctness): PR 已合并但该反馈未在提供的材料中闭环；`apply_block_scaled_mm` 仍为 `raise NotImplementedError`，且该类被放在 ROCm 候选列表首位，走通用 `block-scaled` 分派的模型会直接启动失败。

- Ruff ISC004: tuple 内未加括号的隐式字符串拼接 (style): 作者以 commit chore: wrap string literals in parens 统一补括号, 消息文本不变。
- CI 负载协调与失败用例归因 (other): AndreasKaratzas 确认失败来自 nightly, 属于既有问题, 不阻塞本 PR。

风险与影响

- 风险:

1. 默认选型优先级被改写 (回归面最大): `_POSSIBLE_FP8_BLOCK_KERNELS[PlatformEnum.ROCM]` 中新内核被放到首位, 只要 AITER + fp8 linear 打开且 (N, K) 命中调优表, ROCm 用户的 FP8 block-scaled 层就会静默切换到新内核——权重布局、量化 scale 布局、bias 路径全部换了一套实现。收益依赖调优表命中, 未命中则退回旧内核, 意味着同一份配置在不同 GPU/CU 数上表现不一致, 回归定位会变难。
2. `apply_block_scaled_mm` 直接 `NotImplementedError`: 类被注册进 block-scaled 候选列表, 却拒绝通用入口。mawong-amd 已实机观测到 DeepSeek-V4 因此无法初始化。虽然失败是响亮的 (不是静默错算), 但对“该模型在 ROCm 上本来能跑”的用户是硬性功能回退, 且 `can_implement` 的调优表门禁无法阻止走该入口的调用方。
3. 权重原地改写与布局耦合: `process_weights_after_loading` 把 `params.weight` 替换成 shuffle 后的副本 (可能有额外显存与一次 `contiguous()` 拷贝), 任何绕过 `apply_weights` 读取 `params.weight` 的路径 (其他量化方法、权重重打包、调试导出) 都会拿到不符合直觉的布局; 同时 e8m0 scale 上采样到 fp32 也增加了 scale 侧显存。
4. 未闭环的正确性反馈: mawong-amd 报告的 V3 TP8 乱码没有在材料中得到解释或复现结论, 而 PR 已合并。若问题真实存在, 属于静默精度回归, 比初始化失败更难发现, GSM8k 单点指标 (0.9477) 不足以覆盖 MLA o-proj 之外的所有 FFN/attention 线性层。
5. 外部 CSV 契约依赖: 选型正确性依赖读取 aiter 的 `AITER_CONFIG_GEMM_A8W8_BLOCKSCALE_BPRESHUFFLE_FILE`, 并按 `gfx / cu_num` 列过滤; `_load_gemm_tuned_configs` 用宽泛 `except` 吞掉所有异常返回空集, 行为是安全的 (退回旧内核), 但 aiter 换列名 / 换文件路径时只会静默失去性能收益, 不会报错, 排障成本高。
6. 无自动化测试: 3 个改动文件全部是源码, 没有单测覆盖 `can_implement` 门禁、权重 shuffle 往返一致性或 `is_blockscale_bpreshuffle_tuned` 的过滤逻辑, 后续 aiter 升级和重构都缺少护栏。
 - 影响: 用户侧: 只影响 ROCm (gfx9, MI300/MI350 级别) 且开启 AITER FP8 线性层的用户, CUDA/CPU/XPU 路径完全不受影响; 对 DeepSeek-V3 类 FP8 block-scaled 模型, TP8+DPA 场景实测 +4.6%~+8.1% QPS, TP8+EP +0~+9%, MLA decode o-proj 单 GEMM 从 52us 降到 21us, TTFT/TPOT 同时改善。代价是部分模型 (V4 之类的调用路径) 可能因 `apply_block_scaled_mm` 未实现而无法启动。系统侧: 改变了 ROCm block-scaled FP8 的默认选型顺序, 并让 (N, K) 与 GPU 型号共同决定走哪条 GEMM 路径, kernel 选择从“平台级”细化为“形状 + 硬件级”; `_load_gemm_tuned_configs` 通用化为后续更多调优表 (如 blockscale per-token、mxfp8) 提供了共用入口。团队侧: 确立了“用独立内核类承载不同权重布局、用 tuned config 而非环境变量做门禁”的写法, 日志里可直接看到所选内核类名, 利于现场排障; 同时把 aiter 调优表检查集中到 `_aiter_ops.py`, 为将来交由 aiter 自身分派预留了收敛

点。

- 风险标记：ROCm 选型优先级变更，复杂权重布局原地改写，`apply_block_scaled_mm` 未实现，缺少自动化测试覆盖，外部正确性反馈未闭环

关联脉络

- PR #56161 [Bugfix][ROCm] Create linear layer biases with `requires_grad=False`: 同属 ROCm 线性层 Parameter/DLPack 约束这条线；本 PR 同样用 `torch.nn.Parameter(shuffled_weight.data, requires_grad=False)` 创建洗牌后的权重，二者共同构成 ROCm 线性层参数创建规范。
- PR #55239 [ROCm][Bugfix] Route GLM-5.3-Flash MTP through ragged sparse MLA: 同为 ROCm 上“用分派谓词 / 门禁决定算子走哪条内核”的模式：那个 PR 是放宽 sparse-MLA 谓词修复静默错算，本 PR 是收紧调优表门禁防止 fallback fault，二者说明 ROCm 后端选型门禁是持续出问题的区域。
- PR #56098 [ROCm][Bugfix][Perf] Tune multi-stream shared experts use; `wvSplitKrc` fixes: 同属 ROCm AITER 算子性能调优线索，都通过 `rocm_aiter_ops/` 内核选型层控制具体实现路径，与本 PR 在调优表与内核注册上的改动属于同一演进方向。
- PR #56228 [PR#56228] (讨论中被 mawong-amd 提及) : mawong-amd 在验证本 PR 时指出该 PR 改了 `sparse-attention indexer` 的调用签名，导致在 main 上复现本 PR 需额外打补丁，属于同期 main 上的另一处破坏性改动。