

PR #42061 完整报告

vllm-project/vllm

[ROCm][Bugfix] Re-tag AITER MoE weights as preshuffled after replace_parameter

合并时间: 2026-05-09 12:42

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/42061>

执行摘要

- 一句话: 修复 AITER MoE 权重 shuffle 标记丢失
- 推荐动作: 值得合并。修复逻辑清晰, 与 Quark MoE 的既有做法一致 (quark_moe.py:1286-1287), 已在真实硬件上验证。建议如果后续有同类 replace_parameter 模式, 可将其封装为通用工具函数。

功能与动机

AITER 的 shuffle 操作虽然正确执行了权重重排, 但随后 `replace_parameter` 构造新的 `nn.Parameter` 时丢失了 `is_shuffled` 属性, 导致运行时内核选择器读取 `getattr(layer.w13_weight, "is_shuffled", False) -> False`, 回退到非预重排 (Nswizzle0) 路径, 无法使用针对预重排权重优化的调优内核, 且每次推理时每个 MoE 层都输出警告。

实现拆解

1. FP8 MoE 方法 (vllm/model_executor/layers/quantization/fp8.py): 在 `_setup_kernel` 的 `replace_parameter` 调用后, 新增条件判断 `if self.fp8_backend == Fp8MoeBackend.AITER: layer.w13_weight.is_shuffled = True; layer.w2_weight.is_shuffled = True`。同时从 `oracle/fp8.py` 导入 `Fp8MoeBackend` 枚举。
2. MXFP4 MoE 方法 (vllm/model_executor/layers/quantization/mx_fp4.py): 在 `GptOssMx_fp4MoEMethod._setup_kernel` 和 `Mx_fp4MoEMethod._setup_kernel` 的 `replace_parameter` 调用后 (bias 处理之前), 新增条件判断 `if self.mx_fp4_backend == Mx_fp4MoeBackend.AITER_MXFP4_BF16: layer.w13_weight.is_shuffled = True; layer.w2_weight.is_shuffled = True`。
3. 未量化 MoE 方法 (vllm/model_executor/layers/fused_moe/unquantized_fused_moe_method.py): 在 `_setup_kernel` 的 `replace_parameter` 调用后、`is_weight_update` 判断之前, 新增条件判断 `if self.unquantized_backend == UnquantizedMoeBackend.AITER: layer.w13_weight.is_shuffled = True; layer.w2_weight.is_shuffled = True`。
4. 所有新增代码都放在 `replace_parameter` 之后, 确保刚创建的 `Parameter` 对象立即被标记。
5. 测试: 通过 DeepSeek-V3.2-Exp w8a8 FP8 在 MI355X 上验证, 警告从每进程 24 条降为 0, 且调优内核正确选择 Nswizzle1 路径。

关键文件:

- `vllm/model_executor/layers/quantization/fp8.py` (模块 量化层; 类别 source; 类型 data-contract) : FP8 MoE 量化方法, 新增 AITER 后端标记, 同时导入 `Fp8MoeBackend` 枚举。
- `vllm/model_executor/layers/quantization/mxftp4.py` (模块 量化层; 类别 source; 类型 data-contract) : MXFP4 MoE 量化方法, `GptOssMxftp4MoEMethod` 和 `Mxftp4MoEMethod` 的 `_setup_kernel` 中各新增一处标记。
- `vllm/model_executor/layers/fused_moe/unquantized_fused_moe_method.py` (模块 MoE 层; 类别 source; 类型 data-contract) : 未量化 MoE 方法, 新增 AITER 后端标记。

关键符号: `Fp8MoEMethod._setup_kernel`, `GptOssMxftp4MoEMethod._setup_kernel`, `Mxftp4MoEMethod._setup_kernel`, `UnquantizedFusedMoEMethod._setup_kernel`

关键源码片段

`vllm/model_executor/layers/quantization/fp8.py`

FP8 MoE 量化方法, 新增 AITER 后端标记, 同时导入 `Fp8MoeBackend` 枚举。

```
# vllm/model_executor/layers/quantization/fp8.py
# ... ( 省略无关代码 ) ...

def _setup_kernel(
    self,
    layer: FusedMoE,
    w13: torch.Tensor,
    w2: torch.Tensor,
    w13_scale: torch.Tensor,
    w2_scale: torch.Tensor,
    w13_input_scale: torch.Tensor | None,
    w2_input_scale: torch.Tensor | None,
) -> None:
    # Shuffle weights to runtime format.
    w13, w2, w13_scale, w2_scale = convert_to_fp8_moe_kernel_format(
        fp8_backend=self.fp8_backend,
        layer=layer,
        w13=w13,
        w2=w2,
        w13_scale=w13_scale,
        w2_scale=w2_scale,
        w13_input_scale=w13_input_scale,
        w2_input_scale=w2_input_scale,
    )

    # Replace parameters with updated versions. Note that this helper
    # function ensures the replacement is compatible with RL weight reloads.
    replace_parameter(layer, "w13_weight", w13)
    replace_parameter(layer, "w2_weight", w2)
    replace_parameter(layer, f"w13_{self.weight_scale_name}", w13_scale)
    replace_parameter(layer, f"w2_{self.weight_scale_name}", w2_scale)
```

```

# AITER backend requires weights to be marked as shuffled.
if self.fp8_backend == Fp8MoeBackend.AITER:
    layer.w13_weight.is_shuffled = True
    layer.w2_weight.is_shuffled = True

self.moe_quant_config = self.get_fused_moe_quant_config(layer)
if self.moe_quant_config:
    assert self.experts_cls is not None
    self.moe_kernel = make_fp8_moe_kernel(
        moe_quant_config=self.moe_quant_config,
        moe_config=self.moe,
        fp8_backend=self.fp8_backend,
        experts_cls=self.experts_cls,
        routing_tables=layer._maybe_init_expert_routing_tables(),
        shared_experts=layer.shared_experts,
    )

```

vllm/model_executor/layers/quantization/mx4p.py

MXFP4 MoE 量化方法，GptOssMx4p4MoEMethod 和 Mx4p4MoEMethod 的 _setup_kernel 中各新增一处标记。

```

# vllm/model_executor/layers/quantization/mx4p.py
# ... ( 省略无关代码 ) ...

```

```

def _setup_kernel(self, layer, w13, w2, w13_scale, w2_scale, w13_bias, w2_bias):
    # ... (convert weight to kernel format) ...
    if self.mx4p4_backend not in TRITON_BACKENDS:
        replace_parameter(layer, "w13_weight", w13)
        replace_parameter(layer, "w2_weight", w2)
        replace_parameter(layer, "w13_weight_scale", w13_scale)
        replace_parameter(layer, "w2_weight_scale", w2_scale)
    else:
        layer.w13_weight = w13
        layer.w2_weight = w2
        self.w13_precision_config = w13_scale
        self.w2_precision_config = w2_scale

    # AITER backend requires weights to be marked as shuffled.
    if self.mx4p4_backend == Mx4p4MoeBackend.AITER_MXFP4_BF16:
        layer.w13_weight.is_shuffled = True
        layer.w2_weight.is_shuffled = True

    if w13_bias is not None and w2_bias is not None:
        replace_parameter(layer, "w13_bias", w13_bias)
        replace_parameter(layer, "w2_bias", w2_bias)
    # ... (rest of method) ...

```

vllm/model_executor/layers/fused_moe/unquantized_fused_moe_method.py

未量化 MoE 方法，新增 AITER 后端标记。

```
# vllm/model_executor/layers/fused_moe/unquantized_fused_moe_method.py
# ... ( 省略无关代码 ) ...

def _setup_kernel(
    self,
    layer: Module,
    w13: torch.Tensor,
    w2: torch.Tensor,
) -> None:
    # Shuffle weights to runtime format.
    w13_new, w2_new = convert_to_unquantized_kernel_format(
        self.unquantized_backend,
        layer=layer,
        w13_weight=w13,
        w2_weight=w2,
    )
    # ... (is_weight_update handling) ...
    replace_parameter(layer, "w13_weight", w13_new, prefer_copy=is_weight_update)
    replace_parameter(layer, "w2_weight", w2_new, prefer_copy=is_weight_update)

    # AITER backend requires weights to be marked as shuffled.
    if self.unquantized_backend == UnquantizedMoeBackend.AITER:
        layer.w13_weight.is_shuffled = True
        layer.w2_weight.is_shuffled = True

    if not is_weight_update:
        # ... (setup kernel) ...
```

评论区精华

Reviewer @dllehr-amd 建议精简注释，认为“AITER backend requires weights to be marked as shuffled”即可，无需大段解释。作者 @maeehart 采纳并修剪了所有四个位置的注释为单行。无其他争议。

- 注释长度 (style): 作者采纳建议，将四个文件中的注释均修剪为单行。

风险与影响

- 风险：低风险。每个修改点仅在特定 AITER 后端分支下执行，不影响其他后端。
is_shuffled 属性仅被 AITER 运行时读取，且数据布局确实已重排，标记正确。主路径（非 AITER 后端）无任何行为变化。未附带自动化测试，但作者提供了手动验证结果。
- 影响：对使用 AITER 后端的 ROCm 用户，消除每次推理时每个 MoE 层的警告，并恢复调优内核选择，可能带来性能提升。对其他平台或后端无影响。
- 风险标记：缺少测试覆盖

关联脉络

- PR #39280 [ROCm][Perf] Add Fused Shared Expert (FSE) support for Qwen3-Next: 同为 ROCm 平台 MoE 相关 PR, 涉及 AITER 后端优化, 本 PR 修复了其中可能存在的标记丢失问题。
- PR #42089 [Bugfix] Fix FlashInfer CUTLASS MXFP4-MXFP8 MoE by restoring swizzled scale: 同为 MoE 量化相关 bugfix, 涉及权重格式和元数据正确性。