PR #50597 完整报告

vllm-project/vllm

[ROCm]Remove special-case SiTU support model-specific gating

合并时间：2026-08-15 02:38

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/50597

执行摘要

- 一句话：移除 K3 SiTU 模型特判，改由 oracle 统一路由
- 推荐动作：值得精读，尤其是维护 ROCm / MXFP4 MoE 路径的工程师。这是理解 vLLM modular kernel oracle 路由机制的好样本：把“模型身份”替换为“激活类型 + 后端能力”两个正交维度。建议关注三点：_supports_activation 与 _supports_quant_scheme 的组合如何驱动 oracle 选路；activation 参数如何贯穿 round-up 与 weight convert；AITER_BF16_FP8_MOE_BOUND 统一设置对既有 AITER 用户的回归风险。若团队在 gfx950 上维护 MXFP4 MoE，建议合入后单独跑一遍 DeepSeekV4 的回归验证。

功能与动机

PR body 明确指出：Mxfp4MoEMethod previously contained a model-specific predicate (_use_k3_situ_aiter) that special-cased the Kimi-K3 SiTU activation, gating three separate code paths, 这 "tied SiTU behavior to a specific model identity rather than to the activation type itself"。目标就是让未来任何使用 SiTU 激活 + MXFP4 权重的 ROCm gfx950 模型 "route correctly without requiring new special-case code"。

实现拆解

1. 能力声明下沉：在 vllm/model_executor/layers/fused_moe/experts/rocm_aiter_moe.py 的 AiterExperts._supports_activation 中加入 MoEActivation.SITU，这是 oracle 后端选择器 is_supported_config 接受 SiTU 模型的前提；同时保留 _supports_quant_scheme 中 kMxfp4Static 仅限 gfx950 且非 gfx1250 的约束，避免把 SiTU 支持误放到 gfx1250。
2. oracle 尺寸对齐：oracle/mxfp4.py 的 mxfp4_round_up_hidden_size_and_intermediate_size 增加 activation 参数，并在 current_platform.is_rocm() 分支内对 AITER_MXFP4_BF16 + (SITU 或 SILU) 使用 128 对齐而非通用 256 对齐。原因是 SiTU FlyDSL 内核自带 padding，按 256 对齐会把 TP8 分片后的 384 中间维度撑大导致 OOM。
3. 权重转换收敛：convert_weight_to_mxfp4_moe_kernel_format 增加 activation 参数；在 AITER_MXFP4_BF16 and not is_gfx1250 分支内先统一设置 os.environ["AITER_BF16_FP8_MOE_BOUND"] = "0"，再对 MoEActivation.SITU 分流到 shuffle_weight_a16w4 / shuffle_scale_a16w4 / e8m0_shuffle 的 AITER 专用布局，其余模型仍走 _shuf_w/_shuf_s。原来的 _convert_k3_situ_weight_to_kernel_format 逻辑整体移入此处。

4. Mxfp4MoEMethod 瘦身：__init__ 删掉 is_k3_situ_aiter 分支，统一调用 select_deepseek_v4_mxfp4_moe_backend(moe)；_setup_kernel 无条件调用统一的 convert_weight_to_mxfp4_moe_kernel_format（透传 activation=self.moe.activation），形状断言则仅在 activation != SITU 时执行；process_weights_after_loading 对所有 MXFP4 路径统一调用 _setup_kernel。_use_k3_situ_aiter、is_k3_situ_aiter、_convert_k3_situ_weight_to_kernel_format 全部删除，同时保留 maybe_roundup_sizes 中 K3 需要的 unpad 逻辑（review 中曾因迁移暂时丢失，已恢复）。
5. 测试与验证配套：没有新增单元测试文件；作者用 lm_eval 在 8xMI325X / gfx950 上对 Kimi-K3 做 gsm8k 验证，结果 0.9651 与原路径 0.9666 基本一致；另有 zzw09773 在 #50817 评论中报告 8xMI325X 实测 39-46 tok/s 单流。CI 通过多轮 Buildkite 验证。

关键文件：

- vllm/model_executor/layers/quantization/mxfp4.py（模块 量化层；类别 source；类型 core-logic；符号 Mxfp4MoEMethod, _setup_kernel, process_weights_after_loading, _use_k3_situ_aiter）：Mxfp4MoEMethod 的核心改造文件：删除 _use_k3_situ_aiter / is_k3_situ_aiter / _convert_k3_situ_weight_to_kernel_format，统一走 oracle 后端选择，并把 activation 透传进权重转换。
- vllm/model_executor/layers/fused_moe/oracle/mxfp4.py（模块 后端路由；类别 source；类型 data-contract；符号 mxfp4_round_up_hidden_size_and_intermediate_size, convert_weight_to_mxfp4_moe_kernel_format）：oracle 是本次重构的汇聚点：mxfp4_round_up_hidden_size_and_intermediate_size 与 convert_weight_to_mxfp4_moe_kernel_format 都新增 activation 参数，SITU 的尺寸对齐与权重 shuffle 在这里完成分流。
- vllm/model_executor/layers/fused_moe/experts/rocm_aiter_moe.py（模块 专家内核；类别 source；类型 data-contract；符号 AiterExperts._supports_activation）：通过给 AiterExperts._supports_activation 增加 SITU，让 oracle 的 is_supported_config 能按激活类型自动准入，这是移除模型特判的关键前提。

关键符号：Mxfp4MoEMethod.init, Mxfp4MoEMethod._setup_kernel, Mxfp4MoEMethod.maybe_roundup_sizes, Mxfp4MoEMethod.process_weights_after_loading, mxfp4_round_up_hidden_size_and_intermediate_size, convert_weight_to_mxfp4_moe_kernel_format, AiterExperts._supports_activation

关键源码片段

vllm/model_executor/layers/quantization/mxfp4.py

Mxfp4MoEMethod 的核心改造文件：删除 _use_k3_situ_aiter / is_k3_situ_aiter / _convert_k3_situ_weight_to_kernel_format，统一走 oracle 后端选择，并把 activation 透传进权重转换。

```
# Mxfp4MoEMethod._setup_kernel（head 版本核心片段）
# 所有 MXFP4 后端统一从这里进入权重转换，SITU 不再单独分流。
def _setup_kernel(
    self,
    layer: RoutedExperts,
```

```
    w13: torch.Tensor,
    w2: torch.Tensor,
    w13_scale: torch.Tensor,
    w2_scale: torch.Tensor,
    w13_bias: torch.Tensor | None = None,
    w2_bias: torch.Tensor | None = None,
) -> None:
    num_experts = self.num_experts
    intermediate_size = self.intermediate_size
    hidden_size = self.hidden_size
    sf_block_size = 32

    # SITU 的 FlyDSL 内核内部自带 padding，因此可以处理原生（非 256 对齐）
    # intermediate 尺寸；所以只有非 SITU 激活才做严格形状断言。
    from vllm.model_executor.layers.fused_moe.activation import MoEActivation

    if self.moe.activation != MoEActivation.SITU:
        assert (w13.dim() == 3
                and w13.shape[0] == num_experts
                and w13.shape[1] == intermediate_size * self.moe.w13_num_shards
                and w13.shape[2] == hidden_size // 2)
        assert (w13_scale.dim() == 3
                and w13_scale.shape[0] == num_experts
                and w13_scale.shape[1] == intermediate_size * self.moe.w13_num_shards
                and w13_scale.shape[2] == hidden_size // sf_block_size)
        assert (w2.dim() == 3
                and w2.shape[0] == num_experts
                and w2.shape[1] == hidden_size
                and w2.shape[2] == intermediate_size // 2)
        assert (w2_scale.dim() == 3
                and w2_scale.shape[1] == hidden_size
                and w2_scale.shape[2] == intermediate_size // sf_block_size)
        if w13_bias is not None:
            assert (w13_bias.dim() == 2
                    and w13_bias.shape[0] == num_experts
                    and w13_bias.shape[1] == intermediate_size * self.moe.w13_num_shards)
        if w2_bias is not None:
            assert (w2_bias.dim() == 2
                    and w2_bias.shape[0] == num_experts
                    and w2_bias.shape[1] == hidden_size)

    # 权重格式转换统一交给 oracle，按 activation 参数在函数内部分流。
    w13, w2, w13_scale, w2_scale, w13_bias, w2_bias = (
        convert_weight_to_mxfp4_moe_kernel_format(
            mxfp4_backend=self.mxfp4_backend,
            layer=layer,
            w13_weight=w13,
            w2_weight=w2,
            w13_weight_scale=w13_scale,
```

```
            w2_weight_scale=w2_scale,
            w13_bias=w13_bias,
            w2_bias=w2_bias,
            _cache_permute_indices=self._cache_permute_indices,
            activation=self.moe.activation,
        )
    )

    # TRITON 后端权重是包装张量，不支持 .detach()，需按后端区分赋值方式。
    is_gfx1250 = False
    if current_platform.is_rocm():
        from vllm.platforms.rocm import on_gfx1250
        is_gfx1250 = on_gfx1250()

    uses_triton_weight_format = self.mxfp4_backend in TRITON_BACKENDS or (
        self.mxfp4_backend == Mxfp4MoeBackend.AITER_MXFP4_BF16 and is_gfx1250
    )
    if not uses_triton_weight_format:
        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

    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)
```

vllm/model_executor/layers/fused_moe/oracle/mxfp4.py

oracle 是本次重构的汇聚点：mxfp4_round_up_hidden_size_and_intermediate_size 与 convert_weight_to_mxfp4_moe_kernel_format 都新增 activation 参数，SITU 的尺寸对齐与权重 shuffle 在这里完成分流。

```
# oracle/mxfp4.py 中 AITER_MXFP4_BF16 权重转换分支（gfx950 且非 gfx1250）
# activation 参数让 SiTU 走专用 A16W4 shuffle，其余模型继续走原有 _shuf_w/_shuf_s。
elif mxfp4_backend == Mxfp4MoeBackend.AITER_MXFP4_BF16 and not is_gfx1250:
    # 在分支入口统一设置该环境变量，避免 SiTU 与既有 AITER 路径行为分叉；
    # AITER 侧依赖它关闭 bf16 激活阈值（见 AITER 上游 TODO）。
    import os

    os.environ["AITER_BF16_FP8_MOE_BOUND"] = "0"

    if activation == MoEActivation.SITU:
        from aiter.utility.fp4_utils import e8m0_shuffle
        from vllm._aiter_ops import rocm_aiter_ops
```

```
    fp4_dtype = torch.float4_e2m1fn_x2
    e8m0_dtype = torch.float8_e8m0fnu
    # a8w4（VLLM_ROCM_USE_AITER_MOE_SITUV2_A8W4=1）使用 gate/up 交错的
    # flydsl 内核；默认 a16w4 保持分离布局，所以 w13 按需交错、w2 不交错。
    guinterleave = rocm_aiter_ops.is_fused_moe_situv2_a8w4_enabled()
    w13 = rocm_aiter_ops.shuffle_weight_a16w4(
        w13_weight.data.view(fp4_dtype), 16, guinterleave
    )
    w2 = rocm_aiter_ops.shuffle_weight_a16w4(
        w2_weight.data.view(fp4_dtype), 16, False
    )
    w13_scale_raw = w13_weight_scale.data.view(e8m0_dtype)
    w2_scale_raw = w2_weight_scale.data.view(e8m0_dtype)
    w13_scale = rocm_aiter_ops.shuffle_scale_a16w4(
        w13_scale_raw.view(-1, w13_scale_raw.shape[-1]),
        num_experts,
        guinterleave,
    )
    w2_scale = e8m0_shuffle(w2_scale_raw.view(-1, w2_scale_raw.shape[-1]))
    # 标记已 shuffle ， 后续 precision config 组装依赖该标记。
    w13.is_shuffled = True
    w2.is_shuffled = True
    return (w13, w2, w13_scale, w2_scale, w13_bias, w2_bias)

# 其余 AITER 模型（如 DeepSeekV4）继续走原有 shuffle 路径。
from aiter.ops.shuffle import shuffle_scale as _shuf_s
from aiter.ops.shuffle import shuffle_weight as _shuf_w

w13_weight = torch.nn.Parameter(
    _shuf_w(w13_weight.data.view(torch.float4_e2m1fn_x2),
            is_guinterleave=True, gate_up=True),
    requires_grad=False,
)
shuffled_w13_scale = _shuf_s(
    w13_weight_scale.reshape(-1, w13_weight_scale.shape[-1]),
    num_experts, True, True,
)
w2_weight = torch.nn.Parameter(
    _shuf_w(w2_weight.data.view(torch.float4_e2m1fn_x2),
            is_guinterleave=True, gate_up=False),
    requires_grad=False,
)
shuffled_w2_scale = _shuf_s(
    w2_weight_scale.reshape(-1, w2_weight_scale.shape[-1]),
    num_experts, True, False,
)
# 后续继续原有返回与精度配置组装。
```

评论区精华

review 中主要的交锋集中在三处：

- dllehr-amd 提醒不要动 gfx1250 条件："Don't change this :) we just added gfx1250 last night"，作者回复 "correcting" 并恢复 and not is_gfx1250() 守卫。
- dllehr-amd 发现 unpad 逻辑在迁移中丢失："we still want to unpad right?"，作者承认 "Yep, lost that in the shuffle."，随后恢复。
- 关于 AITER_BF16_FP8_MOE_BOUND 的设置位置，dllehr-amd 担心原来 K3 分支短路的 case 不会触发新设置，建议无条件置 0 并补一次快速测试；作者最终把 oracle 里已有的设置移到 SiTU 条件之上，统一在 AITER_MXFP4_BF16 分支入口生效。
- 另有对 rocm_aiter_moe.py 中多余 __init__ / self.is_situ 的质疑，作者直接删除。
- 保留 gfx1250 条件 (correctness): 恢复 not is_gfx1250() 守卫，gfx1250 不走 AITER_MXFP4_BF16 分支。
- maybe_roundup_sizes 的 unpad 逻辑 (correctness): 恢复 unpad 逻辑，保证 SITU 路径在 round-up 后仍能去掉 padding。
- AITER_BF16_FP8_MOE_BOUND 设置位置 (design): 在 AITER_MXFP4_BF16 分支入口无条件设置环境变量，避免 SiTU 与既有路径行为分叉。
- rocm_aiter_moe 中多余的 init/ is_situ (design): 删除未使用的派生状态，保持 AiterExperts 纯静态能力声明。

风险与影响

- 风险：
  1. 环境变量影响面扩大：AITER_BF16_FP8_MOE_BOUND=0 现在在 convert_weight_to_mxfp4_moe_kernel_format 的 AITER_MXFP4_BF16 分支入口无条件设置，不再只作用于 K3 SiTU。reviewer 也明确提到需要另一个快速测试确认对 DeepSeekV4 等既有 AITER 用户保持关闭状态，这是合入后最值得回归的点。
  2. 形状断言放宽：_setup_kernel 对 SITU 跳过 w13/w2 及 scale 的形状断言。虽然 SiTU 内核支持非 256 对齐的 native 尺寸，但一旦权重形状异常，错误会被推迟到 kernel 运行期而不是加载期暴露，排障成本上升。
  3. 对齐分支移动：SILU 的 128 对齐从顶层分支移入 current_platform.is_rocm() 分支；若未来在非 ROCm 平台使用 AITER_MXFP4_BF16，对齐行为会从 128 变为默认值。当前 AITER 基本只面向 ROCm，实际影响有限。
  4. 缺少直接单元测试：SITU 与其余模型的转换分流、round-up 分支都没有新增单测覆盖，后续改动容易静默回归。 - 影响：对 Kimi-K3 用户：gfx950 + MXFP4 依然走 AITER A16W4 路径，行为不变，且未来新 SiTU 模型不再需要模型补丁。对 DeepSeekV4 等既用 AITER_MXFP4_BF16 的模型：唯一行为差异是环境变量可能更早被设置为 0，以及 round-up 分支位置调整，需要在 gfx950 上做一次精度 / 性能回归。对 ROCm/AITER 开发团队：后端路由收敛到 oracle 单点，is_supported_config 成为唯一准入标准，降低按模型打补丁的维护负担。对上游社区：为其他 SiTU 架构模型铺平了 MXFP4 + AITER 的支持路径。 - 风险标记：核心量化路径重构，环境变量影响面扩大，形状断言

放宽，缺少直接单元测试

关联脉络

- PR #50817 （材料中未提供标题）：zzw09773 在 issue 评论中说明该 PR 与 #50817 组合在 8xMI325X 上验证 Kimi-K3 正确服务（39-46 tok/s 单流，native AITER a16w4 路径）；是同一 gfx950 + SiTU + MXFP4 功能线的配套改动。
- PR #50487 [Model][Spec Decode] Tap the pre-norm AttnRes mixture as the Kimi K3 DFlash aux state: 同为 Kimi-K3 在 vLLM 上的支持工作，涉及 kimi_k3 模型与 ROCm/AITER 运行时；本 PR 则把 K3 的 MXFP4 SiTU 特判从量化层移除，两者共同构成 K3 的 gfx950 部署栈。