

PR #50582 完整报告

vllm-project/vllm

[ROCm][Kimi-K3] aiter moe environment variable cleanup

合并时间: 2026-08-04 02:54

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/50582>

执行摘要

- 一句话: Kimi-K3 AITER MoE 环境变量清理与热路径缓存重构
- 推荐动作: 值得精读。重点看两处设计: 一是 `mxfp4_round_up_hidden_size_and_intermediate_size` 通过可选 `activation` 参数把 K3 特判下沉到 `oracle`, 而不是在调用方保留 `is_k3_situ_aiter` 分支; 二是 `_convert_k3_situ_weight_to_kernel_format` 把 `preshuffle` 从职责过重的 `_setup_kernel_k3_situ` 中剥离, 让 K3 与其他后端共享 `shape` 断言、参数替换与 `kernel` 构建。建议 ROCm 基础设施负责人关注后续 `--moe-backend` 迁移 PR, 并补充针对 `refresh_env_variables` 与 `is_fused_moe_situv2_a8w4_enabled` 的单元测试。

功能与动机

PR body 说明这是对已批准 PR #50162 的 rebase 更新, 核心目标有三: 一是将 `AITER_SITUV2_A8W4` 重命名为 `VLLM_ROCM_USE_AITER_MOE_SITUV2_A8W4`, 遵循 vLLM 命名约定、避免与 AITER 原生 flag 混淆; 二是按 issue #17067 指出的 `envs.ENVIRON` 访问开销 (约 1 微秒 / 次) 把环境变量读取移出 `forward` 热路径, 改在模块加载时缓存; 三是删除无用的 `AITER_SITUV2_A8W4` 导出——AITER 只在 `SEPARATED` 分支读取该变量, 而 vLLM 的 `a8w4` 路径走 `INTERLEAVE`, `GateMode` 已足以选择 `fp8` 内核, 用户无需关心 AITER 内部 flag。

实现拆解

1. 环境变量重命名与热路径缓存: 在 `vllm/envs.py` 中把 `AITER_SITUV2_A8W4` 重命名为 `VLLM_ROCM_USE_AITER_MOE_SITUV2_A8W4` (默认 `False`), 并更新注释说明这是用户唯一需要的开关。在 `vllm/_aiter_ops.py` 中新增类变量 `_MOE_SITUV2_A8W4` 缓存该开关、新增查询函数 `is_fused_moe_situv2_a8w4_enabled()` (要求 `is_fused_moe_enabled()` 同时为真), 并接入 `refresh_env_variables()` 以便测试 `monkey patch` 后刷新。
2. 移除 `forward` 内的 `envs` 读取: `vllm/model_executor/layers/fused_moe/experts/rocm_aiter_moe.py` 删除 `import vllm.envs as envs`, `rocm_aiter_fused_experts()` 中 `SITU` 分支的 `gate_mode` 判定从 `envs.AITER_SITUV2_A8W4` 改为 `rocm_aiter_ops.is_fused_moe_situv2_a8w4_enabled()`, 使热路径命中缓存。
3. AITER workaround 收口到初始化: `vllm/model_executor/layers/quantization/mx_fp4.py` 的 `Mx_fp4MoEMethod.__init__` 中, 当 `is_fused_moe_situv2_a8w4_enabled()` 为真时设置 `os.environ["AITER_BF16_FP8_MOE_BOUND"] = "0"`, 仅限定 `a8w4` 分支且只在 `init` 执

行，避免 forward 开销，并留 TODO 待 AITER 支持 kernel 参数后移除。

4. round-up 特判下沉到 oracle: `vllm/model_executor/layers/fused_moe/oracle/mx_fp4.py` 的 `mx_fp4_round_up_hidden_size_and_intermediate_size()` 新增可选参数 `activation: MoEActivation | None`，对 `AITER_MXFP4_BF16 + SITU` 组合对齐到 128（对 K3 是 no-op，但避免通用 ROCm 256 对齐导致权重膨胀 OOM）；
`Mx_fp4MoEMethod.maybe_roundup_sizes()` 删除 `is_k3_situ_aiter` 特判分支，统一走 oracle。
5. K3 setup 收敛进通用流程: `process_weights_after_loading()` 不再提前 return 到 `_setup_kernel_k3_situ`；`create_weights()` 对 K3 分支调用新的 `_convert_k3_situ_weight_to_kernel_format(layer)` 仅返回四个转换后的张量，随后与通用路径共享 shape 断言、`replace_parameter`、`is_shuffled` 标记、quant config 与 `make_mx_fp4_moe_kernel`。
6. 测试与配套：本 PR 未附带自动化测试文件改动；PR body 提供了 a8w4/a16w4 两条手工验证命令、GSM8K 准确率对比（94.62% vs 95.00%，在噪声范围内）以及 8k/1k con64 性能对比（a8w4 比 a16w4 高 3.5%）。

关键文件：

- `vllm/model_executor/layers/quantization/mx_fp4.py`（模块 MoE 量化；类别 source；类型 data-contract；符号 `_setup_kernel_k3_situ`, `_convert_k3_situ_weight_to_kernel_format`, `Mx_fp4MoEMethod`, `process_weights_after_loading`）：核心量化方法文件：K3 a8w4 workaround 收口到 init，删除 `_setup_kernel_k3_situ` 独立分支，新增 `_convert_k3_situ_weight_to_kernel_format` 使 K3 与其他后端共享通用 `_setup_kernel` 流程。
- `vllm/model_executor/layers/fused_moe/oracle/mx_fp4.py`（模块 内核编排；类别 source；类型 data-contract；符号 `mx_fp4_round_up_hidden_size_and_intermediate_size`）：后端 round-up 逻辑的统一入口：新增 `activation` 参数，把 K3 SITU 特判从 `Mx_fp4MoEMethod` 下沉到 oracle，避免调用方保留 `is_k3_situ_aiter` 分支。
- `vllm/_aiter_ops.py`（模块 AITER 封装；类别 source；类型 core-logic；符号 `is_fused_moe_situv2_a8w4_enabled`, `refresh_env_variables`）：AITER 操作封装的总入口：缓存新环境变量并提供 `is_fused_moe_situv2_a8w4_enabled()`，决定后续 `gate_mode` 与权重布局选择。
- `vllm/model_executor/layers/fused_moe/experts/rocm_aiter_moe.py`（模块 MoE 内核；类别 source；类型 core-logic；符号 `rocm_aiter_fused_experts`）：forward 热路径中的内核分发点：SITU `gate_mode` 判定从 `envs` 读取改为查询缓存，是该 PR 热路径优化的直接受益文件。
- `vllm/envs.py`（模块 环境变量；类别 source；类型 configuration）：环境变量契约文件：定义 `VLLM_ROCM_USE_AITER_MOE_SITUV2_A8W4` 及默认值，直接决定用户配置兼容性。

关键符号：`is_fused_moe_situv2_a8w4_enabled`, `mx_fp4_round_up_hidden_size_and_intermediate_size`, `_convert_k3_situ_weight_to_kernel_format`, `maybe_roundup_sizes`, `refresh_env_variables`, `rocm_aiter_fused_experts`

关键源码片段

vllm/model_executor/layers/quantization/mxfp4.py

核心量化方法文件：K3 a8w4 workaround 收口到 init，删除 _setup_kernel_k3_situ 独立分支，新增 _convert_k3_situ_weight_to_kernel_format 使 K3 与其他后端共享通用 _setup_kernel 流程。

```
# Mxfp4MoEMethod.__init__ 中 a8w4 分支初始化收口：
if self.is_k3_situ_aiter:
    self.mxfp4_backend = Mxfp4MoeBackend.AITER_MXFP4_BF16
    self.experts_cls = backend_to_kernel_cls(self.mxfp4_backend)[0]
    logger.info_once("Using AITER_MXFP4_BF16 for Kimi-K3 SiTU MXFP4 MoE.")
    from vllm._aiter_ops import rocm_aiter_ops

    if rocm_aiter_ops.is_fused_moe_situv2_a8w4_enabled():
        # AITER 在 token 数低于阈值时仍走 bf16 激活，这与 a8w4 路径
        # 调优使用的 fp8 内核不匹配；a16w4 路径不会读取该变量。
        # TODO: AITER 支持以 kernel 参数传入后移除该 workaround。
        os.environ["AITER_BF16_FP8_MOE_BOUND"] = "0"

# K3 preshuffle 独立为纯转换函数，不再承担 setup 职责：
def _convert_k3_situ_weight_to_kernel_format(
    self, layer: RoutedExperts
) -> tuple[torch.Tensor, torch.Tensor, torch.Tensor, torch.Tensor]:
    # K3 的 AITER A16W4 内核使用 separated 的 [gate_all, up_all] 布局，
    # 与 gpt-oss/DeepSeek 路径的 interleaved 布局不同，需在此预打乱。
    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
    num_experts = layer.w13_weight.shape[0]

    # a8w4 (VLLM_ROCM_USE_AITER_MOE_SITUV2_A8W4=1) 使用 gate/up-interleaved
    # 的 fp8 flydsl 内核，需要 w13 权重与 scale 为 interleaved 布局；
    # 默认 a16w4 保持 separated 布局。
    guinterleave = rocm_aiter_ops.is_fused_moe_situv2_a8w4_enabled()
    w13 = rocm_aiter_ops.shuffle_weight_a16w4(
        layer.w13_weight.data.view(fp4_dtype), 16, guinterleave
    )
    w2 = rocm_aiter_ops.shuffle_weight_a16w4(
        layer.w2_weight.data.view(fp4_dtype), 16, False
    )
    w13_scale_raw = layer.w13_weight_scale.data.view(e8m0_dtype)
    w2_scale_raw = layer.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]))
# 只返回转换结果，后续 replace_parameter、is_shuffled 标记与
# make_mxfp4_moe_kernel 由通用 _setup_kernel 统一处理。
return w13, w2, w13_scale, w2_scale
```

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

后端 round-up 逻辑的统一入口：新增 activation 参数，把 K3 SiTU 特判从 Mxfp4MoEMethod 下沉到 oracle，避免调用方保留 is_k3_situ_aiter 分支。

```
def mxfp4_round_up_hidden_size_and_intermediate_size(
    backend: Mxfp4MoeBackend,
    hidden_size: int,
    intermediate_size: int,
    activation: MoEActivation | None = None,
) -> tuple[int, int]:
    """按 backend 要求对 hidden_size 与 intermediate_size 做对齐。"""
    if backend == Mxfp4MoeBackend.AITER_MXFP4_BF16 and activation == MoEActivation.SITU:
        # K3 的 AITER A16W4 SiTU 内核可直接处理 K3 原生 intermediate size
        # (moe_intermediate 3072, TP8 下每分区 384)。对齐到 128 对 K3
        # 形状是 no-op，但能避免通用 ROCm 256 对齐造成权重膨胀与 OOM。
        intermediate_size = round_up(intermediate_size, 128)
        hidden_size = round_up(hidden_size, 128)
    elif backend == Mxfp4MoeBackend.EMULATION:
        # 模拟后端没有 kernel tile，只需满足 OCP MX block 对齐。
        intermediate_size = round_up(intermediate_size, OCP_MX_BLOCK_SIZE)
        hidden_size = round_up(hidden_size, OCP_MX_BLOCK_SIZE)
    # 其余 backend 分支 (DEEPGEMM、MARLIN、TRTLLM、ROCm、CPU) 保持不变
```

vllm/_aiter_ops.py

AITER 操作封装的总入口：缓存新环境变量并提供 is_fused_moe_situv2_a8w4_enabled(), 决定后续 gate_mode 与权重布局选择。

```
class rocm_aiter_ops:
    # 模块导入时一次性读取并缓存环境变量，避免 forward 热路径上反复访问
    # envs.ENVIRON (单次访问约 1 微秒，见 issue #17067)。
    # TODO: 后续将 _MOE_SITUV2_A8W4 收敛到 --moe-backend 统一开关。
    _MOE_SITUV2_A8W4 = envs.VLLM_ROCM_USE_AITER_MOE_SITUV2_A8W4

    @classmethod
    @if_aiter_supported
    def is_fused_moe_situv2_a8w4_enabled(cls) -> bool:
        # a8w4 是 AITER fused MoE 的一个变体，因此同时要求 fused MoE 开关开启。
        return cls.is_fused_moe_enabled() and cls._MOE_SITUV2_A8W4
```

评论区精华

唯一实质 review 来自维护者 tjtananaa 的批准评论：

LGTM. @hongxiayang But we have to have a follow up PR soon, to remove this environment variable by using the `--moe-backend` instead. This PR is temporary to standardized the parameter names according to vLLM convention as to not confused with AITER flag.

要点：本次命名标准化只是临时措施，长期方向是把 AITER 变体选择收进 `--moe-backend` 参数体系，后续必须提交迁移 PR 移除 `VLLM_ROCM_USE_AITER_MOE_SITUV2_A8W4`。

claude[bot] 仅提示 fork 自动 review 被禁用，无技术内容。

- 跟进移除环境变量并改用 `--moe-backend` (design): PR 获批合并，但约定后续必须提交迁移 PR 移除 `VLLM_ROCM_USE_AITER_MOE_SITUV2_A8W4`。
- fork 自动 review 被禁用 (other): 由维护者 tjtananaa 人工 review 并批准。

风险与影响

• 风险：

1. 配置兼容性变化：`vllm/envs.py` 移除了 `AITER_SITUV2_A8W4`，依赖旧变量名的启动脚本或 `recipe` 会静默回退到默认 `a16w4` 路径（不会报错），用户必须改用 `VLLM_ROCM_USE_AITER_MOE_SITUV2_A8W4`。
1. 核心路径变更：`vllm/model_executor/layers/quantization/mx_fp4.py` 删除 `_setup_kernel_k3_situ` 并合并进 `_setup_kernel`，K3 权重转换现在必须满足通用 shape 断言与参数替换流程；`maybe_roundup_sizes` 从 `no-op` 改为 128 对齐，若未来 K3 变体形状变化可能触发不同的 padding 行为，需关注 TP 分片后的尺寸。
2. 缺少自动化测试覆盖：5 个文件变更均无对应测试，`a8w4` 路径依赖手工验证（GSM8K 与 benchmark），CI 无法拦截环境变量误用或 `gate_mode` 选择错误。
3. 依赖 AITER 内部行为：`AITER_BF16_FP8_MOE_BOUND=0` 是 AITER workaround，AITER 升级后该变量语义可能变化；PR 已留 TODO，但需在 AITER 修复后及时移除，避免长期隐性依赖。
4. 热路径缓存一致性：`_MOE_SITUV2_A8W4` 在模块导入时固定，`refresh_env_variables()` 虽已同步更新，但单测中 monkey patch 后若不刷新仍会读到旧值（这是既有机制的已知约束，非新增风险）。- 影响：用户影响：仅影响 Kimi-K3 + ROCm + AITER MoE 用户。启用 `a8w4` 高性能路径时必须改用新环境变量；默认 `a16w4` 路径无感。`recipe` 中不再需要手动设置 `AITER_BF16_FP8_MOE_BOUND`，AITER 内部 flag 对用户隐藏。

系统影响：forward 热路径减少一次 `envs.ENVIRON` 读取（`gate_mode` 判定改查缓存），符合 issue #17067 的方向；K3 的 MoE 初始化与权重转换与其他 MXFP4 后端共享同一 `_setup_kernel` 主干，减少维护分支。

团队影响：为后续 `--moe-backend` 统一开关铺路；ROCm/AITER 维护者需跟进 tjtananaa 提出的后续迁移 PR，并考虑补充新环境变量的单元测试。

- 风险标记：配置兼容性变化，核心路径变更，缺少测试覆盖，依赖 AITER 内部行为

关联脉络

- PR #50162 (原始 PR, 标题未知): 本 PR 正是该已批准 PR 的 rebase 版, PR body 明确引用其链接。
- PR #50656 [Kimi-K3] Add option to shard the shared expert instead of replicating: 同一 Kimi-K3 模型功能线, 且都涉及 AITER/Rocm MoE 行为开关, 后续可能统一迁移到 --moe-backend。
- PR #50383 Shard the K3 Latent-MoE up-projection on large batches: 同为 K3 ROCm/AITER MoE 性能优化路径, 共享 mxfp4 与 rocm_aiter 相关代码。
- PR #50678 K3: Move LatentMoERunner: K3 模型目录重构, 与本 PR 收敛 K3 设置流程的方向一致。