PR #46545 完整报告

vllm-project/vllm

[ROCm] [MoE] [Perf] Shared-expert fusion for bias-routed MoE; enable on MiniMax-M3 mxfp8 model

合并时间：2026-06-26 22:05

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/46545

执行摘要

- 一句话：融合 MiniMax-M3 共享专家到 routed grouped MoE，解码性能提升 5-30%
- 推荐动作：该 PR 值得仔细阅读，尤其关注以下设计决策：
  1. 后端正交设计：通过 aiter 操作 or 环境变量 解耦融合与具体后端，是良好的抽象。
  2. 数值等价性权重计算：shared_expert_weight = 1/routed_scaling_factor 保证缩放后贡献匹配。
  3. 隐含 bug 修复：在 mxfp8_native_moe.py 中调整 bin 基数，避免融合后 id 越界。
  4. Review 演化：从新增参数 /env 到简化设计并复用现有设施，反映良好协作。

功能与动机

MiniMax-M3 在每个 MoE 层都有一个共享专家作为独立的密集 MLP（gate_up GEMM + activation + down GEMM，每次需要单独 launch），60 层下 launch 开销在中低并发时成为主要瓶颈。PR 描述指出 'Folding it into the routed grouped GEMM removes those per-layer launches, which is the dominant cost at low/medium concurrency (decode is launch-bound)'。该融合数值上与独立 MLP 路径等价，确保精度不变。

实现拆解

1. 模型层开关检测：在 vllm/models/minimax_m3/amd/model.py 新增 _fuse_shared_experts_enabled 函数，判断 ROCm 平台、配置含 n_shared_experts、环境变量 VLLM_ROCM_USE_AITER_FUSION_SHARED_EXPERTS 已设置且未启用 expert parallelism。
2. MoE 层构造函数调整：在 MiniMaxM3MoE.__init__ 中，当 fuse_shared_experts=True 时不创建独立的 self.shared_experts MLP，而是将 n_shared_experts 传递给 FusedMoE，并调整 get_expert_mapping 为共享专家创建参数映射。
3. 专家计数与路由开关扩展：在 vllm/model_executor/layers/fused_moe/layer.py 的 determine_expert_counts 中增加对 VLLM_ROCM_USE_AITER_FUSION_SHARED_EXPERTS 环境变量的感知，实现后端正交的融合开关。FusedMoE 构造函数新增 shared_expert_weight 计算：当 apply_routed_scale_to_output 时，权重设为 1/routed_scaling_factor 以使 runner 缩放后共享贡献为 1.0。

4. Router 追加共享 expert slots：在 FusedTopKBiasRouter._compute_routing 中，top-k 选择并重归一化后，为共享 expert 生成 id（从 global_num_experts 开始）和权重，追加到 topk_ids 和 topk_weights，使 grouped GEMM 一并计算。
5. 修复按专家数 binning：在 fused_moe_mxfp8_native 中将 moe_align_block_size 的 bin 基数从 global_num_experts 改为 w13.shape[0]，避免融合后 weight tensor 行数增多导致 id 超出范围；EP 时继续使用 global_num_experts。
6. Router 工厂传递参数：在 router_factory.py 的 create_fused_moe_router 中传递 num_fused_shared_experts 和 shared_expert_weight 给 FusedTopKBiasRouter。

无单独测试文件，但作者提供了详细的本地精度和性能测试结果。

关键文件：

- vllm/models/minimax_m3/amd/model.py（模块 模型定义；类别 source；类型 data-contract；符号 _fuse_shared_experts_enabled）：新增 _fuse_shared_experts_enabled 控制融合开关，修改 MiniMaxM3MoE.__init__ 和 get_expert_mapping 以实现共享专家融合。
- vllm/model_executor/layers/fused_moe/layer.py（模块 MoE 层；类别 source；类型 data-contract；符号 determine_expert_counts, FusedMoE）：核心修改 determine_expert_counts 支持后端正交融合开关；在 FusedMoE 中计算 shared_expert_weight 保证数值等价性。
- vllm/model_executor/layers/fused_moe/router/fused_topk_bias_router.py（模块 路由层；类别 source；类型 data-contract；符号 FusedTopKBiasRouter）：在 FusedTopKBiasRouter 中追加共享 expert slots，是融合的核心逻辑实现。
- vllm/model_executor/layers/fused_moe/experts/mxfp8_native_moe.py（模块 专家执行；类别 source；类型 data-contract；符号 fused_moe_mxfp8_native）：修复融合后 moe_align_block_size 的 binning 基数，避免共享 expert id 超出范围。
- vllm/model_executor/layers/fused_moe/router/router_factory.py（模块 路由工厂；类别 source；类型 data-contract；符号 create_fused_moe_router）：传递 num_fused_shared_experts 和 shared_expert_weight 给 FusedTopKBiasRouter。

关键符号：_fuse_shared_experts_enabled, determine_expert_counts, FusedTopKBiasRouter._compute_routing, FusedMoE.init, fused_moe_mxfp8_native

关键源码片段

vllm/models/minimax_m3/amd/model.py

新增 _fuse_shared_experts_enabled 控制融合开关，修改 MiniMaxM3MoE.__init__ 和 get_expert_mapping 以实现共享专家融合。

```
# vllm/models/minimax_m3/amd/model.py

def _fuse_shared_experts_enabled(config: PretrainedConfig) -> bool:
    """判断是否将共享专家融合进 routed grouped MoE。
    ROCm 仅。通过 ``VLLM_ROCM_USE_AITER_FUSION_SHARED_EXPERTS`` 选择加入
     （router-append 的融合方式独立于 aiter master switch，
```

```
    在 triton/flydsl mxfp8 MoE 上也能工作）；
    需要模型配置中有共享专家，且不在 expert parallelism 模式下
    （EP 会使用 expert_map，本融合暂不支持）。
    """
    return bool(
        current_platform.is_rocm()
        and getattr(config, "n_shared_experts", None)
        and envs.VLLM_ROCM_USE_AITER_FUSION_SHARED_EXPERTS
        and not get_current_vllm_config().parallel_config.enable_expert_parallel
    )

# 在 MiniMaxM3MoE.__init__ 中：
# ... 之前加载 gate 等 ...
self.fuse_shared_experts = _fuse_shared_experts_enabled(config)
# 当融合启用时，不再创建独立的 shared_experts MLP，
# 而是由 FusedMoE 内部处理共享专家（作为 routed expert slot 追加）。
if self.n_shared_experts and not self.fuse_shared_experts:
    self.shared_experts = MiniMaxM3MLP(
        config=config,
        intermediate_size=config.intermediate_size * self.n_shared_experts,
        ...
    )
# 在 FusedMoE 创建时传递 n_shared_experts，使 fused MoE 能够为共享专家
# 分配参数 slot（权重矩阵行数 = routed + n_shared）。
self.experts = FusedMoE(
    ...
    n_shared_experts=(self.n_shared_experts if self.fuse_shared_experts else None),
    ...
)
```

vllm/model_executor/layers/fused_moe/layer.py

核心修改 determine_expert_counts 支持后端正交融合开关；在 FusedMoE 中计算 shared_expert_weight 保证数值等价性。

```
# vllm/model_executor/layers/fused_moe/layer.py

def determine_expert_counts(
    num_experts: int,
    num_redundant_experts: int,
    n_shared_experts: int | None,
    is_act_and_mul: bool,
) -> tuple[int, int, int]:
    global_num_experts = num_experts + num_redundant_experts
    logical_num_experts = num_experts
    # 共享 expert 融合：将共享 expert(s) 作为 routed expert slot 追加，
    # 使其在同一个 grouped GEMM 中执行。通过环境变量控制：
    # - aiter fused-MoE 路径需要 env + master switch 同时打开
    #   (is_fusion_moe_shared_experts_enabled 检查)
    # - 后端正交的 router-append 路径只依赖 env 变量
```

```
    # （例如 MiniMax-M3 的 triton/flydsl mxfp8 MoE）。
    fuse_shared_enabled = (
        rocm_aiter_ops.is_fusion_moe_shared_experts_enabled()
        or envs.VLLM_ROCM_USE_AITER_FUSION_SHARED_EXPERTS
    ) and is_act_and_mul # 仅 gated activation 支持

    num_fused_shared_experts = (
        n_shared_experts if n_shared_experts is not None and fuse_shared_enabled else 0
    )
    return global_num_experts, logical_num_experts, num_fused_shared_experts

# 在 FusedMoE 构造函数中调用此函数获取 num_fused_shared_experts，
# 然后计算 shared_expert_weight 传递给 router：
shared_expert_weight = (
    (1.0 / routed_scaling_factor)
    if (
        apply_routed_scale_to_output
        and num_fused_shared_experts > 0
        and routed_scaling_factor
    )
    else 1.0
)
# 原因：当 apply_routed_scale_to_output 为 True 时，runner 会将
# 所有输出的加权和乘以 routed_scaling_factor；共享 expert 的权重
# 必须设为 1/routed_scaling_factor，使得最终共享贡献被缩放回 1.0
# （与未融合的独立 MLP 加和结果一致）。
```

vllm/model_executor/layers/fused_moe/router/fused_topk_bias_router.py

在 FusedTopKBiasRouter 中追加共享 expert slots，是融合的核心逻辑实现。

```
# vllm/model_executor/layers/fused_moe/router/fused_topk_bias_router.py

class FusedTopKBiasRouter(BaseRouter):
    def __init__(
        self,
        ...,
        num_fused_shared_experts: int = 0,
        shared_expert_weight: float = 1.0,
    ):
        ...
        # 保存融合参数，在 _compute_routing 中使用
        self.num_fused_shared_experts = num_fused_shared_experts
        self.shared_expert_weight = shared_expert_weight

    def _compute_routing(self, hidden_states, router_logits, indices_type, **kwargs):
        # 先计算正常的 top-k routing（已重归一化并乘 routed_scaling_factor）
        topk_weights, topk_ids = fused_topk_bias(...)

        # 如果配置了共享 expert 融合，追加 constant slots
```

```
        if self.num_fused_shared_experts > 0:
            m = topk_ids.shape[0]
            n = self.num_fused_shared_experts
            # global_num_experts 仅计数 routed experts;
            # fused 共享 expert 的 id 紧跟在 routed 之后,
            # 即 [global_num_experts, global_num_experts + n)
            base = self.global_num_experts
            shared_ids = torch.arange(
                base, base + n, dtype=topk_ids.dtype, device=topk_ids.device
            ).expand(m, n)
            shared_w = torch.full(
                (m, n),
                self.shared_expert_weight,
                dtype=topk_weights.dtype,
                device=topk_weights.device,
            )
            topk_ids = torch.cat([topk_ids, shared_ids], dim=-1)
            topk_weights = torch.cat([topk_weights, shared_w], dim=-1)

        return topk_weights, topk_ids
```

评论区精华

- 参数设计简化（@tjtanaa）：建议不新增 fuse_shared_experts bool 参数，通过 n_shared_experts 是否为 None 判断。作者接受并移除。
- 环境变量复用（@tjtanaa）：建议复用现有 VLLM_ROCM_USE_AITER_FUSION_SHARED_EXPERTS，不引入新变量。最终 determine_expert_counts 同时检查 aiter 操作和环境变量。
- CI 测试覆盖（@fxmarty-amd）：指出新增 env 路径未在 CI 中测试。@tjtanaa 回应 CI 资源限制，依赖贡献者本地验证，结果已提供。
- 模型间兼容性（@fxmarty-amd）：担心融合假设所有专家相同精度。确认 MXFP4 和 MXFP8 均一致；默认关闭，当前安全。
- fuse_shared_experts 参数冗余 (design): 作者删除该参数，逻辑移到 determine_expert_counts 内部通过 n_shared_experts 是否为 None 判断。
- 复用现有环境变量代替新建 (design): 最终 determine_expert_counts 同时检查 aiter 操作和环境变量，实现了后端正交的融合开关。
- 缺少 CI 测试覆盖 (testing): tjtanaa 解释 CI 无法运行完整模型测试，依赖贡献者本地验证；已提供详细结果。
- 与 MXFP4 权重的兼容性 (correctness): 确认 MXFP4 的 routed 和 shared expert 精度一致；功能默认关闭，当前安全。

风险与影响

- 风险：

    - 精度风险：共享 expert 权重缩放 1/routed_scaling_factor 仅在 apply_routed_scale_to_output 时有效，若未来缩放策略改变，需重新验证。仅对 MiniMax-M3 MXFP8 路径测试，其他模型需谨慎。
    - 兼容性风险：仅支持 ROCm 平台与 gated activation，NVIDIA 或 CPU 上忽略。若强行开启可能失败。
    - 权重加载风险：当共享专家与 routed 专家量化精度不同时，融合可能导致类型错误。未添加保护，但功能默认关闭。
    - 测试风险：CI 无 end-to-end 测试，回归靠人工。
- 影响：
    - 用户影响：MiniMax-M3 用户可通过设置 VLLM_ROCM_USE_AITER_FUSION_SHARED_EXPERTS=1 获得解码性能提升（低并发 5-30%，高并发 5-10%），精度无损。默认关闭，不影响现有。
    - 系统影响：修改了 MoE 路由核心和专家计数逻辑，其他使用 FusedTopKBiasRouter 的模型未来可复用此模式。
    - 团队影响：提供可复用的共享 expert fusion 模式（router 追加 slots + 权重缩放），降低后续类似优化门槛。
    - 风险标记：核心路径变更，缺少 CI 测试覆盖，精度风险（权重缩放），兼容性（精度不一致风险）

关联脉络

- PR #46419 [ROCm]Enable AITER MoE backend for MiniMax-M3-MXFP4: 该 PR 为 MiniMax-M3 启用 AITER MoE 后端，本 PR 在此基础上进一步实现共享专家融合，且共用相同的环境变量控制。
- PR #44667 [NVFP4][Emulation] Fuse NVFP4 weight dequantization with compute in triton kernel for w13/w2 MOE MLP linears: 同样属于 MoE 融合优化（NVFP4 反量化与计算融合），体现了 vLLM 在 MoE 融合优化方向的持续演进。