PR #23760 完整报告

sgl-project/sglang

[MoE] Unify DeepEPMoE+MoriEPMoE through AITER MoeRunner pre/post-permute

合并时间：2026-05-17 17:25

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23760

执行摘要

- 一句话：统一 DeepEPMoE 与 MoriEPMoE 的 AITER 调用路径，移除 MoriEPMoE 类
- 推荐动作：建议团队中关注 MoE 和 AMD 后端的同学精读此 PR，尤其关注 AiterRunnerCore 的设计和 pre_permute/post_permute 的注册机制，作为后端抽象模式的参考范例。同时建议在合并后尽快在 AMD CI 中添加覆盖各 a2a 后端的测试。

功能与动机

根据 PR 描述，这是 #23597（AITER MoE runner 重构）的后续。之前重构已让大部分量化方法通过 MoeRunner 调用 aiter.fused_moe，但 DeepEPMoE.forward_aiter 和 MoriEPMoE.run_moe_core 两个直接调用点仍需统一。该 PR 旨在消除这些残留直接调用，统一所有 EP 层的执行路径，简化代码结构。

实现拆解

1. AITER 后端管道化：在 moe_runner/aiter.py 中新增 AiterRunnerCore 类（继承 MoeRunnerCore），替代原有的基于 fused_func 注册的风格。注册了 deepep_normal、deepep_ll 和 standard 三种 pre_permute 函数，以及对应的 post_permute 函数。pre_permute 负责处理 DeepEP 的 -1 topk_ids 重路由、mori 的 token 截断和量化类型决议，并将所需信息存入 running_state 字典。post_permute 根据 running_state 中的标志返回对应的 CombineInput 子类（如 DeepEPNormalCombineInput 或 MoriEPNormalCombineInput）。
2. 删除 MoriEPMoE 类：在 ep_moe/layer.py 中移除整个 MoriEPMoE 类，并将 get_moe_impl_class 改为对所有 a2a 后端（包括 mori）返回 DeepEPMoE。DeepEPMoE.__init__ 在启用 AITER 时将 deprecate_flag 直接设为 True，从而跳过 DeepEP 特定初始化，后续 run_moe_core 直接调用父类的标准路径。移除了 forward_aiter 方法。
3. 分发器更新：在 token_dispatcher/deepep.py 和 moriep.py 的 __init__ 中添加 expert_mask_gpu 属性，其形状分别为 (num_local_experts + 1,) 和 (num_experts,)，供 AITER pre_permute 使用。MoriEPDispatcher.combine 方法增加了 hidden_states[:self._num_tokens] 切片，避免返回缓冲区的多余 token。
4. Runner 构造简化：在 moe_runner/runner.py 中，MoeRunner.__init__ 在检测到 AITER 后端时直接 self.runner_core = AiterRunnerCore(config)，替代之前的 side-effect 导入和 runner_core = None 的 fallback。

5. 工具与量化方法适配：在 utils.py 的 MoeA2ABackend 枚举中添加 supports_aiter() 方法，返回支持的 a2a 后端列表。多个量化方法文件（unquant.py、mxfp4.py、fp8.py、quark_w4a4_mxfp4_moe.py、quark_int4fp8_moe.py）移除了对 aiter 的直接引用或条件分支，统一通过 quant_method.apply 经 runner 执行。

关键文件：

- python/sglang/srt/layers/moe/moe_runner/aiter.py（模块 MOE Runner；类别 source；类型 core-logic；符号 AiterRunnerInput, AiterRunnerOutput, _aiter_activation, _aiter_quant_type）：引入了 AiterRunnerCore、AiterRunnerInput / Output 数据类以及 pre/post-permute 注册函数，是统一调度的核心实现文件。
- python/sglang/srt/layers/moe/ep_moe/layer.py（模块 EP 层；类别 source；类型 core-logic；符号 DeepEPMoE.init, DeepEPMoE.run_moe_core, MoriEPMoE (deleted)）：删除了 MoriEPMoE 类，简化 DeepEPMoE 初始化与 run_moe_core，是统一后的主要受益文件。
- python/sglang/srt/layers/moe/token_dispatcher/moriep.py（模块 令牌分发器；类别 source；类型 core-logic；符号 MoriEPDispatcher.init, MoriEPDispatcher.combine）：新增 expert_mask_gpu 初始化，修改 combine 方法以支持 token 切片，适配 AITER 预置变换。
- python/sglang/srt/layers/moe/token_dispatcher/deepep.py（模块 令牌分发器；类别 source；类型 core-logic；符号 DeepEPDispatcher.init）：新增 expert_mask_gpu 初始化，形状为 num_local_experts+1，用于掩码 AITER 预置变换的 sink 槽位。
- python/sglang/srt/layers/moe/utils.py（模块 工具函数；类别 source；类型 core-logic；符号 MoeA2ABackend.supports_aiter）：新增 MoeA2ABackend.supports_aiter() 方法，为统一路由提供后端兼容性查询。

关键符号：AiterRunnerCore.run, DeepEPMoE.init, DeepEPMoE.run_moe_core, MoriEPDispatcher.combine, MoeA2ABackend.supports_aiter

关键源码片段

python/sglang/srt/layers/moe/moe_runner/aiter.py

引入了 AiterRunnerCore、AiterRunnerInput / Output 数据类以及 pre/post-permute 注册函数，是统一调度的核心实现文件。

```
# AiterRunnerCore 是 MoeRunnerCore 的子类，负责驱动 aiter.fused_moe
class AiterRunnerCore(MoeRunnerCore):
    def run(
        self,
        runner_input: AiterRunnerInput,
        quant_info: AiterMoeQuantInfo,
        running_state: dict,
        hooks: Optional[Any] = None,
    ) -> AiterRunnerOutput:
        # 确保配置允许无融合模式未支持
        assert not self.config.no_combine, "no_combine=True is not supported by AITER"
```

```
        # 空输入直接返回，避免 kernel launch
        if runner_input.hidden_states.shape[0] == 0:
            return AiterRunnerOutput(hidden_states=runner_input.hidden_states)

        from aiter.fused_moe import fused_moe

        # 使用 per-token activation scale（mori 提供）或 fallback 到全局 scale
        a1_scale = (
            runner_input.a1_scale
            if runner_input.a1_scale is not None
            else quant_info.a13_scale
        )

        # 为 mori 特有参数收集 extra kwargs
        extra: dict = {}
        if runner_input.num_local_tokens is not None:
            extra["num_local_tokens"] = runner_input.num_local_tokens
        if runner_input.output_dtype is not None:
            extra["dtype"] = runner_input.output_dtype

        output = fused_moe(
            hidden_states=runner_input.hidden_states,
            w1=quant_info.w13_weight,
            w2=quant_info.w2_weight,
            topk_weight=runner_input.topk_weights,
            topk_ids=runner_input.topk_ids,
            quant_type=_aiter_quant_type(runner_input.quant_type),
            activation=_aiter_activation(self.config.activation),
            w1_scale=quant_info.w13_scale,
            w2_scale=quant_info.w2_scale,
            a1_scale=a1_scale,
            a2_scale=quant_info.a2_scale,
            bias1=quant_info.b13,
            bias2=quant_info.b2,
            expert_mask=quant_info.expert_mask,
            doweight_stage1=quant_info.doweight_stage1,
            hidden_pad=quant_info.hidden_pad,
            intermediate_pad=quant_info.intermediate_pad,
            **extra,
        )
        return AiterRunnerOutput(hidden_states=output)
```

python/sglang/srt/layers/moe/ep_moe/layer.py

删除了 MoriEPMoE 类，简化 DeepEPMoE 初始化与 run_moe_core，是统一后的主要受益文件。

```
# DeepEPMoE.__init__ 中 deprecate_flag 的设置，决定是否跳过 DeepEP 特定初始化
class DeepEPMoE(FusedMoE):
    def __init__(self, ...):
```

```
        super().__init__(...)
        # 当启用 AITER 时，直接标记 deprecate_flag 为 True，
        # 意味着后续初始化流程完全跳过 DeepEP 专用逻辑，
        # 后续 run_moe_core 会调用父类的标准 runner 路径
        if _use_aiter:
            self.deprecate_flag = True
        elif _is_npu:
            self.deprecate_flag = False
        elif deep_gemm_wrapper.ENABLE_JIT_DEEPGEMM and isinstance(
            quant_config, Fp8Config
        ):
            self.deprecate_flag = True
        elif (
            deep_gemm_wrapper.ENABLE_JIT_DEEPGEMM
            and envs.SGLANG_DEEPEP_BF16_DISPATCH.get()
        ):
            self.deprecate_flag = True
        else:
            self.deprecate_flag = False
        # ... 后续省略原有 DeepEP 初始化代码
```

评论区精华

审阅者 gemini-code-assist[bot] 提出了几点改进建议：

- expert_mask 数据类型一致性：建议 deepep.py 中 torch.int 改为 torch.int32，与 moriep.py 保持一致。
- 导入位置优化：建议将 AiterRunnerCore.run 中的 aiter 模块导入移到类 __init__ 中，避免每次调用重复导入。
- torch.where 优化：建议使用标量 runner_config.num_local_experts 替代 torch.full_like 创建的全量张量。 这些建议未在本次 PR 中得到明确采纳或回复，PR 已合并，可能留待后续优化。
- expert_mask 数据类型建议统一为 int32 (style): 未明确采纳，PR 已合并时 deepep.py 仍使用 torch.int。
- AiterRunnerCore 中 aiter 导入建议放到 __init__中避免重复导入 (performance): 未在本次 PR 中实施，可能作为后续优化。

风险与影响

- 风险：
  - 废弃路径：MoriEPMoE 类被完全删除，所有 mori 调用转由 DeepEPMoE 处理。若 mori 存在未被覆盖的特定行为（如特殊的环境变量或内存布局），可能导致功能回归。
  - AMD 硬件依赖：AITER 后端仅对 HIP （AMD GPU）生效。修改涉及所有 a2a 后端（deepep、mooncake、nixl、mori），需要在 AMD 硬件上全面验证正确性和性能。
  - 性能风险：新增的 Python 层 pre/post-permute 查找和数据类构造在每次 MoE 前向调用时执行，可能引入微秒级额外开销。作者认为无变化，但建议在 MI300X 上做基准对比。

    - 环境变量行为变化：SGLANG_USE_AITER 以前只影响部分量化方法，现在全面控制 AITER 后端，可能影响依赖特定行为的用户。
- 影响：
    - 用户角度：统一接口降低认知复杂度，但 AMD 用户可能需要重新验证模型精度和吞吐。mori 用户不再感知单独的 MoriEPMoE 类，所有日志和调试信息将引用 DeepEPMoE。
    - 系统角度：删除约 280 行代码，新增约 100 行，整体简化。减少了硬编码分支，增加了多态间接层（MoeRunnerCore），但为未来添加新 a2a 后端提供了更清晰的扩展点。
    - 团队角度：降低维护成本，新开发者只需遵循 pre/post-permute 模式即可集成新后端。
    - 风险标记：废弃 MoriEPMoE 路径，AMD 硬件验证，环境变量行为变化，性能影响待验证

关联脉络

- PR #23597 AITER MoE runner refactor: 该 PR 的基础，提供了 MoeRunner 框架和 fused_func 注册机制，本 PR 在其上进一步统一。