PR #40817 完整报告

vllm-project/vllm

[Models] Cohere MoE

合并时间：2026-04-29 23:54

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/40817

执行摘要

- 一句话：新增 Cohere MoE 模型支持，含自定义路由 SigmoidRenorm
- 推荐动作：建议阅读该 PR 以理解 Cohere MoE 的实现方式，特别是自定义路由函数的设计和枚举扩展模式。对于使用 Cohere MoE 的用户，等待模型权重发布后再进行验证。注意审查残差连接和 RoPE 的实现是否符合预期。

功能与动机

根据 PR 描述，目的是将 Cohere MoE 模型集成到 vLLM 中，模型权重即将可用。用户可通过 `vllm serve` 启动推理服务，使用 OpenAI 兼容 API 进行在线聊天。

实现拆解

1. 核心模型实现：新增 vllm/model_executor/models/cohere_moe.py，定义路由函数 token_choice_with_bias（sigmoid + topk + renormalize）、共享专家 MLP CohereMoeMLP、注意力层 CohereMoeAttention、解码层 CohereMoeDecoderLayer 和整体模型 CohereMoeModel，并注册为 CohereMoeForCausalLM。
2. 扩展自定义路由路由器：在 vllm/model_executor/layers/fused_moe/router/custom_routing_router.py 的 routing_method_type 属性中增加对 token_choice_with_bias 的识别，返回新枚举值 RoutingMethodType.SigmoidRenorm。
3. 路由配置更新：插入 SigmoidRenorm 枚举值并调整 Custom、Simulated 和 Unspecified 的数值以保持与 FlashInfer 后端的顺序一致；同时在 trtllm_nvfp4_moe.py 后端加入对该路由类型的支持。
4. 模型注册：在 vllm/model_executor/models/registry.py 的 _TEXT_GENERATION_MODELS 字典中添加 "CohereMoeForCausalLM": ("cohere_moe", "CohereMoeForCausalLM")。
5. 多模态模型调整：修改 vllm/model_executor/models/cohere2_vision.py，导入 SupportsQuant 接口并在 Cohere2VisionForConditionalGeneration 的基类中增加 SupportsQuant；添加 packed_modules_mapping 以支持权重合并；移除 hf_to_vllm_mapper 中过时的 lm_head 前缀映射。
6. 测试配套：更新 tests/models/registry.py，添加 CohereMoeForCausalLM 的 HuggingFace 示例信息（指向内部路径，is_available_online=False）。

7. 文档更新：在 docs/models/supported_models.md 中将 Cohere MoE 加入支持的模型列表。

关键文件：

- vllm/model_executor/models/cohere_moe.py（模块 模型执行器；类别 source；类型 data-contract；符号 token_choice_with_bias, CohereMoeMLP, init, forward）：新增的 Cohere MoE 模型主文件，包含自定义路由函数、MLP、注意力、解码层和整体模型类，是本次变更的核心。
- vllm/model_executor/models/cohere2_vision.py（模块 多模态模型；类别 source；类型 data-contract；符号 Cohere2VisionForConditionalGeneration）：修改多模态视觉模型以支持量化接口（SupportsQuant）并添加权重打包映射，使 Cohere MoE 文本模型与视觉模型共享量化能力。
- vllm/model_executor/layers/fused_moe/router/custom_routing_router.py（模块 MoE 路由；类别 source；类型 data-contract）：扩展自定义路由路由器，使其能识别 Cohere MoE 的新路由函数 token_choice_with_bias 并返回 SigmoidRenorm 类型。
- vllm/model_executor/models/registry.py（模块 注册表；类别 source；类型 data-contract）：在模型注册表中添加 CohereMoeForCausalLM 的映射，使其能被 vLLM 识别和加载。
- vllm/model_executor/layers/fused_moe/experts/trtllm_nvfp4_moe.py（模块 MoE 后端；类别 source；类型 data-contract）：在 nvfp4 MoE 后端中添加对 SigmoidRenorm 路由类型的支持，确保 Cohere MoE 可使用该量化后端。
- tests/models/registry.py（模块 测试注册表；类别 test；类型 test-coverage）：添加 CohereMoeForCausalLM 的测试注册信息，确保模型能在测试框架中被识别。
- docs/models/supported_models.md（模块 文档；类别 docs；类型 documentation）：更新支持的模型列表，将 Cohere MoE 加入文档中，方便用户查阅。

关键符号：token_choice_with_bias, CohereMoeModel.forward, CustomRoutingRouter.routing_method_type

关键源码片段

vllm/model_executor/models/cohere_moe.py

新增的 Cohere MoE 模型主文件，包含自定义路由函数、MLP、注意力、解码层和整体模型类，是本次变更的核心。

```
@torch.compile(backend=current_platform.simple_compile_backend)
def token_choice_with_bias(
    hidden_states: torch.Tensor,
    gating_output: torch.Tensor,
    topk: int,
    renormalize: bool,
):
    # Sigmoid -> top-k (-> renormalize) 自定义路由，用于 Cohere MoE
    assert hidden_states.shape[0] == gating_output.shape[0], "Token 数量不匹配"
```

```
    # 对门控输出应用 sigmoid 获得分数
    scores = gating_output.float().sigmoid()
    # 选择 top-k 个专家的权重与索引
    topk_weights, topk_ids = torch.topk(scores, k=topk, dim=-1, sorted=False)

    if renormalize:
        # 对 top-k 权重进行归一化（除以和）
        topk_weights = topk_weights / topk_weights.sum(dim=-1, keepdim=True)

    return topk_weights.to(torch.float32), topk_ids.to(torch.int32)


class CohereMoeMLP(nn.Module):
    # Cohere MoE 中使用的共享专家 MLP
    def __init__(
        self,
        config: CohereConfig,
        intermediate_size: int | None = None,
        quant_config: QuantizationConfig | None = None,
        prefix: str = "",
    ):
        super().__init__()
        self.config = config
        self.hidden_size = config.hidden_size
        self.intermediate_size = (
            intermediate_size if intermediate_size is not None else config.intermediate_size
        )
        # 合并的 gate + up 投影
        self.gate_up_proj = MergedColumnParallelLinear(
            self.hidden_size,
            [self.intermediate_size] * 2,
            bias=False,
            quant_config=quant_config,
            prefix=f"{prefix}.gate_up_proj",
        )
        self.down_proj = RowParallelLinear(
            self.intermediate_size,
            self.hidden_size,
            bias=False,
            quant_config=quant_config,
            reduce_results=False,
            prefix=f"{prefix}.down_proj",
        )
        self.act_fn = SiluAndMul()

    def forward(self, x):
        # 执行 gate_up -> SiLU 激活 -> down 投影
        gate_up, _ = self.gate_up_proj(x)
        x = self.act_fn(gate_up)
```

```
        x, _ = self.down_proj(x)
        return x
```

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

扩展自定义路由路由器，使其能识别 Cohere MoE 的新路由函数 token_choice_with_bias 并返回 SigmoidRenorm 类型。

```
class CustomRoutingRouter(BaseRouter):
    # ... ( 部分代码省略 ) ...

    @property
    def routing_method_type(self) -> RoutingMethodType:
        # 延迟导入以避免循环依赖
        from vllm.model_executor.models.cohere_moe import token_choice_with_bias
        from vllm.model_executor.models.llama4 import Llama4MoE

        # Llama4 使用 FlashInfer TRTLLM 后端
        if self.custom_routing_function == Llama4MoE.custom_routing_function:
            return RoutingMethodType.Llama4
        # Cohere MoE 使用 sigmoid -> top-k -> renormalize 路由
        if self.custom_routing_function == token_choice_with_bias:
            return RoutingMethodType.SigmoidRenorm
        return RoutingMethodType.Custom

    def _compute_routing(
        self,
        hidden_states: torch.Tensor,
        router_logits: torch.Tensor,
        indices_type: torch.dtype | None,
        *,
        input_ids: torch.Tensor | None = None,
    ) -> tuple[torch.Tensor, torch.Tensor]:
        # 使用自定义路由函数计算路由
        topk_weights, topk_ids = self.custom_routing_function(
            hidden_states=hidden_states,
            gating_output=router_logits,
            topk=self.top_k,
            renormalize=self.renormalize,
        )
        return topk_weights.to(torch.float32), topk_ids.to(
            torch.int32 if indices_type is None else indices_type
        )
```

评论区精华

RoPE 应用范围：gemini-code-assist 指出 RoPE 应应用于所有注意力层而非仅滑动窗口层，否则位置信息丢失会导致错误。作者回复 "This is intentional"，认为当前设计符合意图。该争议未导致代码修改，最终合并。

残差连接逻辑：gemini-code-assist 指出 CohereMoeDecoderLayer 中残差连接逻辑错误：先覆盖了前层残差变量，后又错误地叠加。作者未直接回复，但最终代码可能已修正（需进一步确认）。

枚举值顺序调整：DarkLight1337 询问改变 RoutingMethodType 枚举值是否安全。mgoin 和 wzhao18 确认新值 SigmoidRenorm 正确，并需与 FlashInfer 保持一致。wzhao18 指出有在飞 PR 也更新相同枚举。最终采用调整后的方案。

- RoPE 应应用于所有注意力层 (correctness): 作者回复 'This is intentional'，认为当前设计符合意图。
- 残差连接逻辑潜在问题 (correctness): 未收到作者直接回复，但最终合并的代码可能已修正，需额外确认。
- 枚举值顺序调整与兼容性 (design): 确认调整可接受，wzhao18 指出有在飞 PR 也更新相同枚举。

风险与影响

- 风险：
  1. 新模型代码稳定性：cohere_moe.py 为全新实现，尚未经过大规模生产验证，可能存在未发现的 bug 或性能问题。
  2. 路由函数正确性：自定义路由 token_choice_with_bias 使用 sigmoid + topk 而非 softmax，需确认与原始 Cohere MoE 模型权重和行为一致。
  3. 残差连接逻辑：review 中指出残差连接可能出错，若未正确修复会影响模型输出。
  4. 枚举兼容性：调整 RoutingMethodType 枚举值可能影响序列化或与 FlashInfer 后端的交互，但已通过讨论确认兼容。
  5. RoPE 覆盖范围：RoPE 仅应用于滑动窗口层，若模型设计依赖全层位置编码，则推理结果可能存在偏差。 - 影响：用户：可加载并使用 Cohere MoE 模型进行推理，体验与标准 OpenAI API 兼容的服务。系统：新增约 550 行代码，但改动集中在模型执行器子目录，不影响现有模型加载、调度或推理核心。团队：需维护新模型实现，后续若模型权重发布，可快速验证。路由枚举的调整可能与其他 MoE 后端（如 FlashInfer）的演进产生联动。
- 风险标记：新模型代码稳定性，路由函数正确性，残差连接需验证，枚举兼容性已确认

关联脉络

- 暂无明显关联 PR