PR #48884 完整报告

vllm-project/vllm

[Model] Add Inkling LoRA support [4/N]

合并时间：2026-07-17 09:52

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/48884

执行摘要

- 一句话：为 Inkling 模型添加完整的 LoRA 适配支持，涵盖注意力、MoE 及 lm_head
- 推荐动作：值得精读。该 PR 展示了在复杂 MoE 架构（共享 sink expert、共享 -outer 布局、muP 缩放）中集成 LoRA 的系统性方法，设计决策（如条件选择线性层、检测包装器分流、共享 expert 维为 1 的存储模型）有参考价值。建议关注 InklingSinkExpertsLinear 的 _gamma_expand 实现和 _lora_forward 的 muP 合成逻辑。由于缺少真实适配器评估，在实际部署前应进行充分测试。

功能与动机

PR 旨在为 Inkling 模型提供 LoRA 微调支持，使其能够加载 LoRA 适配器。如 PR Body 所述：'Mark Inkling as LoRA-capable and map adapter names for its packed QKVR projection, shared sink experts, and LM head.' 同时需要支持 Inkling 独特的共享 -outer MoE 适配器布局和 muP 输出缩放。Lamport 融合收集路径直接访问权重会绕过 LoRA delta，因此要求设置环境变量 INKLING_MULTIMEM_AR=0。

实现拆解

1. 启用 LoRA 支持与适配器映射：在 vllm/models/inkling/nvidia/model.py 中让 _TmlForCausalLMBase 继承 SupportsLoRA，定义 packed_modules_mapping（将 qkvr 映射到 wq_du、wk_dv 等，将 w13 映射到 w1、w3）和 embedding_modules（将 lm_head 映射到 output_embeddings），以及 hf_to_vllm_mapper 中增加前缀映射，使 LoRA 管理器能正确识别 Inkling 的权重结构。
2. 共享 Sink Expert 的 LoRA 兼容层：在 vllm/models/inkling/nvidia/moe.py 中新增 InklingSinkExpertsLinear 类，使用 MergedColumnParallelLinear 和 RowParallelLinear 替代原先的 fused 实现。Forward 中实现 gate-up 拆解、SiLU 激活、gamma 扩展和输出投影。在 InklingMoE 的 __init__中根据 lora_config 动态选择 InklingSinkExpertsLinear 或原 InklingSinkExperts。
3. Logits 处理器的 muP 与 LoRA 合成：重写 vllm/models/inkling/nvidia/logits_processor.py 的 forward，通过检测 self.base_layer 是否存在来分流到 _lora_forward 或 _base_forward。_base_forward 保持不变（muP 折叠进 GEMM alpha），_lora_forward 调用包装器的 _get_logits 获得 base+LoRA logits 后整体乘以 1/mup。
4. FusedMoEWithLoRA 的共享 -outer 布局支持：在 vllm/lora/layers/fused_moe.py 中添加 enable_moe_shared_loras 属性。新增 _w13_a_num_experts 和 _w2_b_num_experts

属性，当启用共享时返回 1。create_lora_weights 中在构建 lora_a_stacked/lora_b_stacked 时使用这些属性选择 expert 索引。在 _build_lora_context 中传递 enable_moe_shared_loras 到 MoELoRAContext。

5. PackedLoRALayerWeights 新增打包方法：在 vllm/lora/lora_weights.py 中添加 pack_moe_stacked 类方法，接受三个预堆叠的 LoRALayerWeights，直接使用它们的 lora_a/lora_b 构造 PackedLoRALayerWeights，支持共享 -outer 布局（expert-dim 为 1）。
6. 配置与引擎集成：在 vllm/config/lora.py 中添加 enable_moe_shared_loras 字段（默认 False），并在 vllm/engine/arg_utils.py 中暴露对应参数。模型管理器在初始化时读取该配置，传递给 process_packed_modules_mapping 和 FusedMoEWithLoRA，并在 _slice_moe_lora_ep 中保护 shared expert 切片。_create_merged_loras_inplace 中当 _enable_moe_shared_loras 时调用 pack_moe_stacked 而非 pack_moe。
7. 测试：修改 tests/models/inkling/test_moe_weight_layout.py，添加 test_gate_is_not_a_lora_target 和 test_custom_embedding_is_not_a_lora_target，验证 gate 和自定义嵌入不被视为 LoRA 目标模块。

关键文件：

- vllm/models/inkling/nvidia/moe.py（模块 模型层；类别 source；类型 data-contract；符号 InklingSinkExpertsLinear, init, _gamma_expand, load_weight）：新增 InklingSinkExpertsLinear 类，提供 LoRA 兼容的共享 sink expert 计算；动态选择线性或融合实现，是 Inkling LoRA 支持的核心。
- vllm/lora/layers/fused_moe.py（模块 LoRA 层；类别 source；类型 core-logic；符号 _w13_a_num_experts, _w2_b_num_experts, _match_expert_dim, enable_moe_shared_loras）：添加共享 -outer MoE 适配器支持，新增 enable_moe_shared_loras 属性及 _w13_a_num_experts/_w2_b_num_experts 属性，修改 create_lora_weights 和 _build_lora_context，是 LoRA 框架的核心改动。
- vllm/models/inkling/nvidia/logits_processor.py（模块 模型层；类别 source；类型 data-contract；符号 _lora_forward, _base_forward）：重写 forward 以支持 LoRA 路径，新增 _lora_forward 和 _base_forward，确保 muP 缩放正确作用于包含 LoRA delta 的完整 logits。
- vllm/lora/model_manager.py（模块 LoRA 层；类别 source；类型 data-contract；符号 _slice_local）：初始化时读取 enable_moe_shared_loras 配置，传递给 process_packed_modules_mapping 和 FusedMoEWithLoRA；修改 _slice_moe_lora_ep 保护 shared expert 切片；在 _create_merged_loras_inplace 中支持 pack_moe_stacked。
- vllm/lora/lora_weights.py（模块 LoRA 层；类别 source；类型 core-logic；符号 pack_moe_stacked）：新增 pack_moe_stacked 类方法，用于打包预堆叠的 expert LoRA 张量，是共享 -outer 布局的关键辅助方法。
- vllm/models/inkling/nvidia/model.py（模块 模型定义；类别 source；类型 data-contract；符号 _TmlForCausalLMBase）：让 _TmlForCausalLMBase 继承 SupportsLoRA，定义 packed_modules_mapping 和 embedding_modules，使 LoRA 管理器能正确识别 Inkling 的权重结构。

- vllm/config/lora.py（模块 配置；类别 source；类型 dependency-wiring）：新增 enable_moe_shared_loras 配置项，控制共享 -outer MoE 适配器的启用，是功能开关的入口。
- tests/models/inkling/test_moe_weight_layout.py（模块 测试；类别 test；类型 test-coverage；符号 test_gate_is_not_a_lora_target, test_custom_embedding_is_not_a_lora_target）：添加测试验证 gate 和自定义嵌入不是 LoRA 目标模块，确保 LoRA 支持的正确范围。

关键符号：InklingSinkExpertsLinear.init, InklingSinkExpertsLinear._gamma_expand, InklingSinkExpertsLinear.load_weight, InklingSinkExpertsLinear.forward, InklingLogitsProcessor.forward, InklingLogitsProcessor._lora_forward, InklingLogitsProcessor._base_forward, FusedMoEWithLoRA._w13_a_num_experts, FusedMoEWithLoRA._w2_b_num_experts, FusedMoEWithLoRA.enable_moe_shared_loras, FusedMoEWithLoRA.create_lora_weights, PackedLoRALayerWeights.pack_moe_stacked

关键源码片段

vllm/models/inkling/nvidia/moe.py

新增 InklingSinkExpertsLinear 类，提供 LoRA 兼容的共享 sink expert 计算；动态选择线性或融合实现，是 Inkling LoRA 支持的核心。

```
class InklingSinkExpertsLinear(nn.Module):
    """LoRA-capable implementation of the Inkling sink experts."""

    def __init__(
        self,
        n_experts: int,
        d_model: int,
        d_mlp: int,
        *,
        prefix: str = "",
    ) -> None:
        super().__init__()
        # 使用 MergedColumnParallelLinear 实现 LoRA 兼容的 w13 投影
        from vllm.model_executor.layers.linear import (
            MergedColumnParallelLinear,
            RowParallelLinear,
        )
        self.n_experts = n_experts
        self.d_mlp = d_mlp
        total = n_experts * d_mlp # 所有 expert 的 MLP 中间维度总和
        self.w13 = MergedColumnParallelLinear(
            input_size=d_model,
            output_sizes=[total, total], # gate 和 up 各占一半
            bias=False,
            prefix=f"{prefix}.w13",
        )
        self.w2 = RowParallelLinear(
```

```
            input_size=total,
            output_size=d_model,
            bias=False,
            reduce_results=False,
            prefix=f"{prefix}.w2",
        )
        self._w2_input_pp = self.w2.input_size_per_partition
        self._col_expert: torch.Tensor | None = None

    def _gamma_expand(self, gammas: torch.Tensor) -> torch.Tensor:
        """将 gammas (batch, n_experts) 展开到 TP 切分后的 w2 输出维度。"""
        if self._col_expert is None or self._col_expert.device != gammas.device:
            local = self._w2_input_pp # 本 rank 的 w2 输入大小
            start = get_tensor_model_parallel_rank() * local
            cols = torch.arange(start, start + local, device=gammas.device)
            # 每个列对应的 expert 索引 : cols // d_mlp
            self._col_expert = (cols // self.d_mlp).long()
        return gammas[:, self._col_expert]

    def load_weight(self, key: str, weight: torch.Tensor) -> list[str]:
        """加载权重，处理 inkling 的交错 gate/up 存储格式。"""
        if key == "w13_weight":
            d_model = weight.shape[-1]
            # 隔行取 gate (0,2,4...) 和 up (1,3,5...)
            gate = weight[:, 0::2, :].reshape(-1, d_model).contiguous()
            up = weight[:, 1::2, :].reshape(-1, d_model).contiguous()
            self.w13.weight_loader(self.w13.weight, gate, 0)
            self.w13.weight_loader(self.w13.weight, up, 1)
            return ["w13.weight"]
        # w2_weight: 从 (n_experts, d_mlp, d_model) 变换为 (d_mlp*n, d_model)
        w = weight.permute(1, 0, 2).reshape(weight.shape[1], -1).contiguous()
        self.w2.weight_loader(self.w2.weight, w)
        return ["w2.weight"]

    def forward(self, x: torch.Tensor, gammas: torch.Tensor) -> torch.Tensor:
        """前向：gate-up 投影 -> SiLU -> 乘以 gamma -> w2 输出。"""
        gate_up, _ = self.w13(x)
        gate, up = gate_up.chunk(2, dim=-1)
        hidden_states = torch.nn.functional.silu(gate) * up
        # gamma_expand 将 (batch, n_experts) 展开到 w2 的 TP 划分维度
        hidden_states = (hidden_states * self._gamma_expand(gammas)).to(x.dtype)
        output, _ = self.w2(hidden_states)
        return output
```

vllm/lora/layers/fused_moe.py

添加共享 -outer MoE 适配器支持，新增 enable_moe_shared_loras 属性及 _w13_a_num_experts/_w2_b_num_experts 属性，修改 create_lora_weights 和 _build_lora_context，是 LoRA 框架的核心改动。

```
class FusedMoEWithLoRA(BaseLayerWithLoRA):
    def __init__(self, base_layer: MoERunner) -> None:
        ...
        # 从 lora_config 设置，默认为 False
        self.enable_moe_shared_loras = False
        ...

    @property
    def _w13_a_num_experts(self) -> int:
        """Expert 维度：共享时为 1，否则为 local_num_experts。"""
        return 1 if self.enable_moe_shared_loras else self.local_num_experts

    @property
    def _w2_b_num_experts(self) -> int:
        """Expert 维度：共享时为 1，否则为 local_num_experts。"""
        return 1 if self.enable_moe_shared_loras else self.local_num_experts

    def _create_lora_a_weights(self, max_loras, lora_config):
        # 使用 _w13_a_num_experts 替代固定 local_num_experts
        self.w13_lora_a_stacked = tuple(
            torch.zeros((max_loras, self._w13_a_num_experts, ...), ...)
            for _ in range(...)
        )

    def create_lora_weights(self, ...):
        self.enable_moe_shared_loras = lora_config.enable_moe_shared_loras
        ...
        for experts_id in range(self.local_num_experts):
            # 共享时 expert 索引固定为 0
            w13_a_eid = 0 if self.enable_moe_shared_loras else experts_id
            w2_b_eid = 0 if self.enable_moe_shared_loras else experts_id
            self.lora_a_stacked.append(
                self.w13_lora_a_stacked[0][lora_id][w13_a_eid]
            )
            ...
```

vllm/models/inkling/nvidia/logits_processor.py

重写 forward 以支持 LoRA 路径，新增 _lora_forward 和 _base_forward，确保 muP 缩放正确作用于包含 LoRA delta 的完整 logits。

```
class InklingLogitsProcessor(LogitsProcessor):
    def forward(self, lm_head, hidden_states, embedding_bias=None):
        # base_layer 仅在 LogitsProcessorWithLoRA 包装器中存在
        # 此时 self 是包装器，需要显式调用基类的 _lora_forward
        if hasattr(self, "base_layer"):
            return type(self.base_layer)._lora_forward(
                self, lm_head, hidden_states, embedding_bias
            )
        return self._base_forward(lm_head, hidden_states, embedding_bias)
```

```
def _lora_forward(self, lm_head, hidden_states, embedding_bias=None):
    # self 是 LogitsProcessorWithLoRA 包装器
    # 先获取 base_logits + LoRA delta
    mup_multiplier = self.base_layer.logits_mup_width_multiplier
    mup = 1.0 / mup_multiplier if mup_multiplier else None
    if self.logits_as_input:
        logits = hidden_states
    else:
        logits = self._get_logits(hidden_states, lm_head, embedding_bias)
    # 对整个 logits 应用 muP 缩放，使 LoRA delta 也被缩放
    if logits is not None and mup:
        assert self.base_layer.soft_cap is None
        assert self.base_layer.scale == 1.0
        logits = logits * mup
    return logits

def _base_forward(self, lm_head, hidden_states, embedding_bias=None):
    mup = self.logits_mup_width_multiplier
    if not mup:
        return super().forward(lm_head, hidden_states, embedding_bias)
    # 将 muP 除数折叠进 GEMM alpha，避免额外 kernel
    assert self.soft_cap is None
    assert self.scale == 1.0
    w = lm_head.weight
    if self._logits_zero is None:
        self._logits_zero = w.new_zeros(1)
    logits = torch.addmm(
        self._logits_zero,
        hidden_states,
        w.t(),
        beta=0.0,
        alpha=1.0 / mup,
    )
    logits = self._gather_logits(logits)
    if logits is not None:
        logits = logits[..., :self.org_vocab_size]
    return logits
```

评论区精华

本 PR 没有产生 Review 评论，但自动审核机器人 claude[bot] 告知需要手动 Review。仓库维护者 jeejeelee 已批准该 PR。PR Body 声明是从草稿 PR #48768 提取的 LoRA 部分，并指出需要最终评估：'A full serving evaluation with a real Inkling adapter remains required before this draft is marked ready.' 没有未解决的讨论。注意：PR 作者强调必须 Review 每一行变更，并完成完整模型评估。

- 暂无高价值评论线程

风险与影响

- 风险：
  - 环境变量约束：要求 INKLING_MULTIMEM_AR=0，如果用户未设置，Lamport 融合收集路径会直接访问权重从而绕过 LoRA delta，导致结果错误或异常。
  - 共享 -outer 布局首次实现：enable_moe_shared_loras 是新提出的概念，可能与其他 MoE 量化或 EP 配置存在交互 bug，测试覆盖有限。
  - muP 缩放逻辑：_lora_forward 中通过 hasattr(self, 'base_layer') 检测包装器，依赖于 LoRA 管理器注入的约定，如果未来 LoRA 包装器实现变化可能失效。
  - 缺少真实适配器评估：PR Body 承认未完成全量服务评估，只运行了单元测试，可能存在运行时未发现的语义错误。
  - 单次 commit：所有变更一次性提交，没有中间审查点，增加了风险集中度。
- 影响：
  - 用户影响：启用 LoRA 功能的 Inkling 模型用户现在可以加载 LoRA 适配器进行微调推理。需要设置 INKLING_MULTIMEM_AR=0 环境变量。性能上，LoRA 激活时额外执行 LoRA 前向，但 PR 尽量通过条件选择保持非 LoRA 场景的零开销。
  - 系统影响：修改了通用 LoRA 框架（FusedMoEWithLoRA、PackedLoRALayerWeights、模型管理器），但这些改动的激活条件仅限于 Inkling 模型（通过 enable_moe_shared_loras 配置），对其他模型无影响。
  - 团队影响：需维护新引入的共享 -outer 布局路径和配置项，后续 Inkling 模型演进需注意兼容性。
  - 风险标记：环境变量约束，共享 -outer 布局首次实现，缺少真实适配器评估，单次 commit 无中间审查

关联脉络

- PR #48869 [Model] Add Inkling MTP=1 support [3/N]: 本 PR 是 Inkling 支持系列 [4/N]，与 [3/N] 紧密关联，两者分别实现 MTP 和 LoRA，共同构成 Inkling 模型的完整适配。
- PR #48858 [Model] Add Hopper FA4 relative attention for Inkling: 同为 Inkling 模型的功能扩展，关注注意力机制。
- PR #48822 [Model] Add PW CUDA graph support for Inkling [2/N]: 同为 Inkling 模型的功能扩展，关注 CUDA 图支持。