PR #23976 完整报告

sgl-project/sglang

Support Gemma3/4 + Eagle3

合并时间：2026-05-10 04:34

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23976

执行摘要

- 一句话：支持 Gemma3/4 模型与 Eagle3 推测解码
- 推荐动作：该 PR 值得精读，特别是 _shard_weight 的实现解决了 TP 环境下 Embedding 共享的关键问题，layers_to_capture 的偏移设计也值得借鉴。对于需要将新模型接入 Eagle3 的开发者，可直接复用 set_eagle3_layers_to_capture 和 get_embed_and_head 等接口。建议在合并后尽快补充测试覆盖捕获路径和分片逻辑。

功能与动机

PR 描述明确说明："This PR supports Gemma3/4 model with Eagel3 and fixes multiple bugs in current eagle3 implementation. - Support aux layers embedding captures for both models. Fixes an issue when trying to capture the last layer - Support an additional norm layer for each aux embedding. In practice, this could help stabilize the training and improve accept rate - Gemma3/4 use nn.Embedding (Gemma3TextScaledWordEmbedding) which is not TP aware, while eagle3 drafter uses VocabParallelEmbedding. When TP>1, eagle3 drafter will get full copy of embedding."

实现拆解

实现分为以下步骤：

1. 在文本模型中增加中间层捕获机制：在 Gemma3TextModel 和 Gemma4TextModel 的 __init__ 中初始化 self.layers_to_capture = []；在 forward 中遍历每一层时，若当前层索引在 layers_to_capture 中，则将输入 hidden_states 追加到 aux_hidden_states，最后在循环后检查 num_layers 是否在列表中以捕获最终层输出。
2. 引入捕获标志和解包：在 Gemma3ForCausalLM 和 Gemma4ForCausalLM 中添加 capture_aux_hidden_states = False，当该标志为 True 时，从 hidden_states 中解包 (hidden_states, aux_hidden_states)，并传递给 LogitsProcessor（已扩展支持 aux_hidden_states 参数）。多模态包装器 Gemma3ForConditionalGeneration 和 Gemma4ForConditionalGeneration 同样添加了该标志和对应的解包逻辑。
3. 实现 set_eagle3_layers_to_capture 方法：每个模型类均实现该方法，默认捕获三层（[2, num_layers // 2, num_layers - 3]），并通过 +1 偏移将用户传入的层编号转换为内部索引（输入缓存对应输出）。

4. 解决 TP>1 嵌入分片不匹配：在 Gemma4TextModel 中新增 _shard_weight 方法，对完整 Embedding 权重按 tensor parallel rank 切分，使得由 get_embed 和 get_embed_and_head 返回的权重与 Eagle3 草稿模型使用的 VocabParallelEmbedding 兼容。Gemma3TextModel 中类似地实现了分片逻辑（_shard_weight）并调整返回。
5. 辅助隐藏状态归一化：在 Eagle3DraftModel（llama_eagle3.py）中添加可选的 use_aux_norm 配置，若启用则在 forward 中将三个辅助隐藏状态分别通过独立的 RMSNorm，然后拼接，以平衡不同层级贡献。

关键文件：

- python/sglang/srt/models/gemma3_causal.py（模块 文本模型；类别 source；类型 data-contract；符号 set_eagle3_layers_to_capture, _shard_weight, get_embed, get_embed_and_head）：核心文本模型，添加 layers_to_capture 和 aux_hidden_states 收集逻辑，新增 set_eagle3_layers_to_capture、_shard_weight、get_embed、get_embed_and_head 方法，并修改 forward 返回值以支持 Eagle3。
- python/sglang/srt/models/gemma4_causal.py（模块 文本模型；类别 source；类型 data-contract；符号 _shard_weight, get_embed, get_embed_and_head, set_eagle3_layers_to_capture）：Gemma4 文本模型，新增 _shard_weight 解决 TP>1 嵌入分片问题，添加 layers_to_capture 和对应方法。
- python/sglang/srt/models/gemma4_mm.py（模块 多模态模型；类别 source；类型 data-contract；符号 get_embed, get_embed_and_head, set_eagle3_layers_to_capture）：多模态包装器，添加 capture_aux_hidden_states 标志并实现 get_embed、get_embed_and_head、set_eagle3_layers_to_capture 委托方法。
- python/sglang/srt/models/gemma3_mm.py（模块 多模态模型；类别 source；类型 data-contract；符号 get_embed_and_head, set_eagle3_layers_to_capture）：多模态包装器，添加 get_embed_and_head 和 set_eagle3_layers_to_capture 委托方法。
- python/sglang/srt/models/llama_eagle3.py（模块 草稿模型；类别 source；类型 data-contract）：Eagle3 草稿模型，添加可选 aux_norm 机制，在 forward 中对辅助隐藏状态进行独立的 RMSNorm，提升聚合稳定性。

关键符号：set_eagle3_layers_to_capture, _shard_weight, get_embed, get_embed_and_head

关键源码片段

python/sglang/srt/models/gemma3_causal.py

核心文本模型，添加 layers_to_capture 和 aux_hidden_states 收集逻辑，新增 set_eagle3_layers_to_capture、_shard_weight、get_embed、get_embed_and_head 方法，并修改 forward 返回值以支持 Eagle3。

```
# Gemma3TextModel 中修改后的 forward 方法
# 添加了 aux_hidden_states 收集和支持返回
def forward(self, input_ids, positions, forward_batch, input_embeds=None, **kwargs):
    if input_embeds is None:
        hidden_states = self.embed_tokens(input_ids)
    else:
```

```
            hidden_states = input_embeds

        aux_hidden_states = [] # 收集指定层的输入（即前一层的输出）
        num_layers = len(self.layers)

        # 每层前先判断是否捕获当前 hidden_states
        for i, layer in enumerate(self.layers):
            if i in self.layers_to_capture:
                aux_hidden_states.append(hidden_states)
            # ... 正常的 layer 前向 ...
            hidden_states = layer(...)[0]

        # 如果配置了捕获最后一层的输出（索引 num_layers），则捕获
        if num_layers in self.layers_to_capture:
            aux_hidden_states.append(hidden_states)

        hidden_states = self.norm(hidden_states)
        if not aux_hidden_states:
            return hidden_states
        return hidden_states, aux_hidden_states

    # 新增 set_eagle3_layers_to_capture 方法
    def set_eagle3_layers_to_capture(self, layer_ids=None):
        if layer_ids is None:
            num_layers = len(self.layers)
            # 默认捕获低、中、高三个层，内部存储偏移 +1
            self.layers_to_capture = [2, num_layers // 2, num_layers - 2]
        else:
            # 用户传入的层编号 +1 偏移，因为捕获的是输入缓存（对应前一层的输出）
            self.layers_to_capture = [i + 1 for i in layer_ids]
```

python/sglang/srt/models/gemma4_causal.py

Gemma4 文本模型，新增 _shard_weight 解决 TP>1 嵌入分片问题，添加 layers_to_capture 和对应方法。

```
    # Gemma4TextModel 中的 _shard_weight 方法
    # 在 TP>1 时将完整 Embedding 权重按 vocab 维度分片，兼容 VocabParallelEmbedding
    def _shard_weight(self, weight: torch.Tensor) -> torch.Tensor:
        tp_size = get_tensor_model_parallel_world_size()
        if tp_size <= 1:
            return weight
        tp_rank = get_tensor_model_parallel_rank()
        shard_size = (weight.shape[0] + tp_size - 1) // tp_size
        # 按 rank 切片
        return weight[tp_rank * shard_size : (tp_rank + 1) * shard_size]

    # get_embed 和 get_embed_and_head 利用 _shard_weight 返回分片权重
    def get_embed(self):
        return self._shard_weight(self.model.embed_tokens.weight)
```

```
def get_embed_and_head(self):
    embed_shard = self._shard_weight(self.model.embed_tokens.weight)
    # weight tying: lm_head 共享 embed_tokens 权重
    return embed_shard, embed_shard
```

评论区精华

Review 讨论主要围绕以下主题：

- embed_scale 处理方式：Reviewer @kpham-sgl 建议将 embed_scale 统一放置于 HF transformer utils 中。作者 @pyc96 经考虑后决定暂时移除 embed_scale 逻辑，等待与 SpecForge 训练侧对齐后再加入。该讨论已达成共识并关闭。
- 空行格式：Reviewer 指出了一个多余空行，作者已删除。
  - embed_scale 处理方式 (design): 移除了 embed_scale 相关代码，未合入本 PR。
  - 空行删除 (style): 作者删除了该空行。

风险与影响

- 风险：
  1. TP>1 嵌入分片正确性：_shard_weight 的计算逻辑必须与 VocabParallelEmbedding 的分片方式一致，否则会导致草稿模型推理错误。该风险存在于 gemma4_causal.py 和 gemma3_causal.py。
  2. forward 返回值变更：当 layers_to_capture 非空时，forward 返回 (hidden_states, aux_hidden_states) 而非单一的 hidden_states。所有调用者（包括 CUDA graph 捕获）必须适应这一变化。当前通过 capture_aux_hidden_states 标志控制，但若其他路径未更新可能导致异常。
  3. aux_norm 配置缺失：use_aux_norm 默认为 False，未显式启用时训练无法受益，且与训练侧的对接尚未完成。
  4. CUDA graph 兼容性：由于 forward 条件分支增加，带捕获路径的 CUDA graph 捕获可能需要额外测试确保图稳定。
- 影响：
  1. 用户影响：使用 Gemma3/4 模型的用户现可启用 Eagle3 推测解码，提升生成速率。TP>1 场景下嵌入分片问题得到修复，Eagle3 草稿模型可正确使用目标模型的嵌入层。
  2. 系统影响：改动仅限于 Gemma3/4 模型相关的 5 个文件，未影响其他后端或模型。
  3. 团队影响：为未来其他模型接入 Eagle3 提供了清晰模式（set_eagle3_layers_to_capture + _shard_weight）。Eagle3 草稿模型的 aux_norm 设计为可选，降低了默认推理的侵入性。 - 风险标记：TP>1 embedding 分片正确性，forward 返回值变更，aux_norm 默认关闭，CUDA graph 兼容性

关联脉络

- PR #24217 fix: STANDALONE spec-decode hidden-size mismatch crash: 该 PR 修复了 Eagle 推测解码的隐藏层大小不匹配问题；本 PR 进一步解决了 Gemma 模型在相同场景下的嵌入层分片问题，两者共同完善了 Eagle3 的模型兼容性。