PR #21885 完整报告

sgl-project/sglang

[LoRA] Torch Native enhancement: embedding and graph optimization

合并时间：2026-05-07 22:28

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/21885

执行摘要

- 一句话：LoRA torch native 后端支持 embedding 并修复 CUDA 图兼容性
- 推荐动作：值得精读，尤其是 __init__.py 中基于 use_cuda_graph 的调度设计，以及 graph_lora_ops.py 中为图兼容而采用的 masking 循环模式。这些是 LoRA 后端与图优化结合的关键技巧，对类似需求有借鉴意义。

功能与动机

根据 issue #20525，torch native LoRA 后端在 CUDA 图模式下生成不正确，导致 LoRA logprob 测试失败。此 PR 修复该问题，同时为 LoRA embedding 层提供支持。

实现拆解

1. 实现 embedding 前向操作：在 lora_ops.py 中添加 sgemm_lora_a_embedding_fwd 函数，支持从权重中直接索引 token 的 embedding，并通过 torch.nn.functional.embedding 查找后乘以 scaling 因子。
2. 创建图兼容版本：新增 graph_lora_ops.py，包含三个图前向函数（sgemm_lora_a_embedding_graph_fwd, sgemm_lora_a_graph_fwd, sgemm_lora_b_graph_fwd），使用 masking 和循环取代控制流中的分段处理，以避免 CUDA 图不支持的条件分支。
3. 重构 init.py 作为调度入口：新的 sgemm_lora_a_fwd、sgemm_lora_b_fwd、sgemm_lora_a_embedding_fwd 函数根据 batch_info.use_cuda_graph 选择调用图版本或控制流版本，对外保持统一接口。
4. 修改 torch_backend.py：新增 run_lora_a_embedding 方法，并将原有多个方法（run_lora_a_sgemm, run_lora_b_sgemm, run_qkv_lora, run_gate_up_lora）的参数由零散的 weight_indices_cpu、seg_lens_cpu 等改为直接接受 batch_info 对象，同时新增 output_offset_cpu 参数用于图捕获。
5. 更新 layers.py：在 VocabParallelEmbeddingWithLoRA、ParallelLMHeadWithLoRA、ReplicatedLinearWithLoRA 中新增 output_offset_cpu 的创建（pin_memory=True）并传递给后端，同时调整 apply_lora 调用以传递新参数。
6. 测试配套：在 test_lora_ops.py 中新增 5 个测试，覆盖 embedding 前向、expand 以及三个图前向函数；更新 lora_utils.py 中 reference_embedding_lora_a_shrink 增加 scaling 参数以对齐实现；test_chunked_sgmv_backend.py 添加一行兼容性改动。

关键文件：

- test/manual/lora/test_lora_ops.py（模块 LoRA 测试；类别 test；类型 test-coverage；符号 test_sgemm_lora_a_embedding_fwd, test_sgemm_lora_a_embedding_fwd_expand, test_sgemm_lora_a_embedding_graph_fwd, test_sgemm_lora_a_graph_fwd）：新增 5 个测试用例，全面覆盖 embedding 和图操作，确保正确性和回归防护。
- python/sglang/srt/lora/backend/torch_backend.py（模块 后端；类别 source；类型 core-logic；符号 run_lora_a_embedding, run_lora_a_sgemm, run_lora_b_sgemm, run_qkv_lora）：实现 run_lora_a_embedding 方法，并重构多个 run_ 方法以支持 batch_info 和 output_offset_cpu，是兼容图的关键。
- python/sglang/srt/lora/torch_ops/graph_lora_ops.py（模块 图操作；类别 infra；类型 infrastructure；符号 sgemm_lora_a_embedding_graph_fwd, sgemm_lora_a_graph_fwd, sgemm_lora_b_graph_fwd）：提供三个图兼容前向函数，使用 masking 循环避免控制流，是支持 CUDA 图的核心。
- python/sglang/srt/lora/torch_ops/__init__.py（模块 调度层；类别 infra；类型 infrastructure；符号 sgemm_lora_a_embedding_fwd, sgemm_lora_a_fwd, sgemm_lora_b_fwd）：提供统一调度入口，根据 use_cuda_graph 切换图或控制流实现，是架构设计的核心。
- python/sglang/srt/lora/layers.py（模块 层；类别 source；类型 core-logic）：在多个 LoRA 层中添加 output_offset_cpu，确保图捕获时偏移量在 CPU 上固定。
- python/sglang/srt/lora/torch_ops/lora_ops.py（模块 操作符；类别 infra；类型 infrastructure；符号 sgemm_lora_a_embedding_fwd）：新增 sgemm_lora_a_embedding_fwd 控制流函数，并重构现有函数使用 addmm_ 等优化。

关键符号：sgemm_lora_a_embedding_fwd, sgemm_lora_a_fwd, sgemm_lora_b_fwd, sgemm_lora_a_embedding_graph_fwd, sgemm_lora_a_graph_fwd, sgemm_lora_b_graph_fwd, run_lora_a_embedding, reference_embedding_lora_a_shrink

关键源码片段

python/sglang/srt/lora/backend/torch_backend.py

实现 run_lora_a_embedding 方法，并重构多个 run_ 方法以支持 batch_info 和 output_offset_cpu，是兼容图的关键。

```
# python/sglang/srt/lora/backend/torch_backend.py ( 修改后 )

def run_lora_a_embedding(
    self,
    input_ids: torch.Tensor,
    weights: torch.Tensor,
    vocab_size: int,
    extra_embeddings: torch.Tensor = None,
    *args,
    **kwargs,
) -> torch.Tensor:
    # 当前 chunked 后端暂不支持 extra_embeddings
```

```
        assert extra_embeddings is None, \
            "Extra embeddings for lora a is not supported yet in chunked backend"
        # 通过 batch_info 判断是否使用 CUDA 图模式，调度到对应实现
        output_tensor = sgemm_lora_a_embedding_fwd(
            inputs=input_ids,
            weights=weights,
            batch_info=self.batch_info,
            vocab_size=vocab_size,
        )
        return output_tensor

    def run_lora_a_sgemm(
        self,
        x: torch.Tensor,
        weights: torch.Tensor,
        stack_num: int = 1,
        *args,
        **kwargs,
    ) -> torch.Tensor:
        # 参数从分散的 cpu 张量简化为 batch_info，后端内部处理设备管理
        output_tensor = sgemm_lora_a_fwd(
            inputs=x,
            weights=weights,
            batch_info=self.batch_info,
            num_slices=stack_num,
        )
        return output_tensor
```

python/sglang/srt/lora/torch_ops/graph_lora_ops.py

提供三个图兼容前向函数，使用 masking 循环避免控制流，是支持 CUDA 图的核心。

```
# python/sglang/srt/lora/torch_ops/graph_lora_ops.py ( 新增 )

def sgemm_lora_a_embedding_graph_fwd(
    inputs: torch.Tensor, # (total_seq_len,) token IDs
    weights: torch.Tensor, # (num_loras, max_rank, vocab_size)
    weight_indices: torch.Tensor, # (total_seq_len,) 每个 token 所属 lora 索引
    seg_len_tensor: torch.Tensor, # (batch_size,) 序列长度
    scaling_tensor: torch.Tensor, # (num_loras,) 缩放因子
    vocab_size: int, # 保留用于未来兼容
) -> torch.Tensor:
    total_seq_len = inputs.shape[0]
    if weights.numel() == 0:
        return torch.zeros(total_seq_len, 0, dtype=weights.dtype, device=weights.device)

    num_loras, max_rank, _ = weights.shape
    output = torch.zeros(total_seq_len, max_rank, dtype=weights.dtype, device=weights.device)

    # 为每个 lora adapter 独立处理：通过 masking 选出属于该 adapter 的 token
```

```
    for lora_idx in range(num_loras):
        batch_token_mask = weight_indices[:total_seq_len] == lora_idx
        x_seq = torch.where(batch_token_mask, inputs, 0) # 不属于的 token 置 0
        w_seq = weights[lora_idx] # (max_rank, vocab_size)
        # 使用 F.embedding 进行查找，然后乘 scaling 并累加
        output.add_(
            scaling_tensor[lora_idx]
            * torch.where(
                batch_token_mask.unsqueeze(1), # 在 rank 维度广播 mask
                F.embedding(x_seq, w_seq.t()), # 查找 embedding 后加 mask
                0,
            )
        )
    return output
```

python/sglang/srt/lora/torch_ops/__init__.py

提供统一调度入口，根据 use_cuda_graph 切换图或控制流实现，是架构设计的核心。

```
# python/sglang/srt/lora/torch_ops/__init__.py（调度入口）

def sgemm_lora_a_fwd(
    inputs: torch.Tensor,
    weights: torch.Tensor,
    batch_info: LoRABatchInfo,
    num_slices: int = 1,
) -> torch.Tensor:
    # 根据图模式选择实现路径
    if batch_info.use_cuda_graph:
        # 图版本：使用 masking 循环，无条件分支
        return sgemm_lora_a_graph_fwd(
            inputs, weights,
            batch_info.weight_indices, # 使用设备上的张量
            batch_info.seg_lens,
            batch_info.scalings,
            num_slices,
        )
    else:
        # 控制流版本：使用分段计算和 addmm
        return sgemm_lora_a_control_fwd(
            inputs, weights,
            batch_info.weight_indices_cpu,
            batch_info.seg_lens_cpu,
            batch_info.lora_ranks_cpu,
            batch_info.scalings_cpu,
            num_slices,
        )
```

评论区精华

review 中 gemini-code-assist[bot] 指出 sgemm_lora_a_embedding_graph_fwd 的 vocab_size 参数未使用，建议移除；作者回应保留以保持未来更新的 API 兼容性。另一评论指出 torch.where 在外层冗余，作者已在 commit 0eefa63 中解决。

- vocab_size 参数未使用 (design): 作者回应保留以保持 API 兼容性，便于未来更新。
- 冗余 torch.where 操作 (performance): 作者通过 commit 0eefa63 解决了该问题。

风险与影响

- 风险：图操作使用 masking 和循环，对长序列或大批量可能带来额外性能开销；保留 vocab_size 参数可能导致接口混淆；控制流与图路径的分离增加了维护成本，但带来了清晰的架构。此外，run_lora_a_embedding 断言 extra_embeddings 为 None，若未来需要扩展 embedding 可能需要修改断言。
- 影响：影响用户：使用 torch_native LoRA 后端的用户现在可以启用 CUDA 图并获得性能提升，同时 embedding 层 LoRA 也可正常工作。影响系统：重构了 API 签名，所有后端调用需适应 batch_info 和 output_offset_cpu，但向后兼容测试未发现回归（已通过 CI 测试）。影响团队：维护者需注意两种路径的同步更新。
- 风险标记：图路径性能开销，API 兼容性冗余参数，extra_embeddings 未实现

关联脉络

- PR #20525 [Bug] Cuda graph with Torch Native LoRA Backend: 此 PR 修复的 issue，motivation 来源于此