

PR #38578 完整报告

sgl-project/sglang

[LoRA] Support MoE in full and breakable prefill CUDA graphs

合并时间: 2026-09-12 05:40

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/38578>

执行摘要

- 一句话: MoE LoRA 支持 full/breakable 预填图, 移除 32 请求上限
- 推荐动作: 值得精读。重点看三个设计决策: (1) decode 与 prefill 的 MoE scratch 分离如何避免图捕获指针冲突; (2) 用请求数作为元数据安全上界的论证; (3) 16-token tile 网格压缩如何在 CUDA grid 上限 (65535) 内覆盖 ragged 请求。若团队维护自定义 LoRA 后端, 需同步更新 `init_prefill_cuda_graph_batch_info` 签名并实现 `max_num_requests` 语义。

功能与动机

PR body 明确提出目标: "Enable MoE LoRA in full and breakable prefill CUDA graphs, including batches above 32 requests." 此前 LoRA 在 prefill CUDA graph 中仅支持 dense 模型且 Triton 后端硬编码了 32 个请求段槽 (`PREFILL_CUDA_GRAPH_LORA_SEGMENTS = 32`), MoE 模型 (Qwen3.5-35B-A3B、Inkling 等) 的 LoRA 只能走 eager prefill, 吞吐受限。该 PR 要解决两个问题: 一是让 MoE 适配器也能进入 prefill 图以获得图执行收益; 二是解除 32 请求限制, 使超过 32 个并发请求的 LoRA 批次不再静默回退 eager。

实现拆解

实现按以下 6 步拆解:

1. 变更入口: 按容量分配 LoRA 元数据。`python/sglang/srt/model_executor/runner/prefill_cuda_graph_runner.py` 的 `__init__` 中 `_capture_lora` 分支改为向 `lora_manager.init_prefill_cuda_graph_batch_info` 传入 `max_num_requests`: Full 后端取 `_capture_req_slots` (即 `full_prefill_max_req`), Breakable 取 `min(self.max_num_tokens, self.max_bs)`; 同时删除原先 "把 Full 请求槽钳制到 LoRA 段槽数" 的日志与钳制逻辑。捕获入口 `run_once` 增加 `model_capture_mode()` 上下文 (仅 Full + LoRA 时启用), 确保 capture 使用纯 base-model 请求时也能把 MoE LoRA kernels 记录进图。
2. MoE scratch 与 decode 分离。`python/sglang/srt/lora/backend/base_backend.py` 的 `init_cuda_graph_moe_buffers` 增加 `prefill: bool = False` 参数, 缓冲区字典按 prefill/decode 分别存入 `prefill_moe_cg_buffers` 或 `moe_cg_buffers`; 同时精简缓冲区集合 (移除 `intermediate_cache1/2/3`、`out_hidden_states`、`weight_indices_long` 等字段)。`_add_moe_lora_info` 通过 `batch_info.is self.prefill_cuda_graph_batch_info` 判断是否为 prefill 图, prefill 时把 "全部请求槽 (含空槽)" 纳入 segment 视图, 因为捕获固定了

align kernel 的段数。_compute_moe_lora_info 新增尾部清理：

token_lora_mapping[num_tokens:].fill_(-1)，防止大 batch replay 后残留旧 adapter 映射污染小 batch。

3. Triton 后端：移除 32 上限 + 16-token tile 网格压缩。

python/sglang/srt/lora/backend/triton_backend.py 删除

PREFILL_CUDA_GRAPH_LORA_SEGMENTS = 32，改为

PREFILL_CUDA_GRAPH_LORA_CHUNK_SIZE = 16。

init_prefill_cuda_graph_batch_info 的槽位数改为 max_num_requests（未传则用 max_num_tokens）；新增 prefill_cuda_graph_sgemm_batch_info，按最坏情况 tile 数（ $\text{ceil}(\text{tokens} / 16) + \text{请求数} - 1$ ）分配段数并 clamp 到 65535（CUDA grid y/z 上限）。prepare_lora_batch 在 prefill 图模式下调用 merge_and_chunk_segments，把各请求 token 按 16-token tile 切分合并，重写 weight_indices 与 seg_lens，清空尾部段后做 cumsum 得到 seg_indptr。

4. CSGMV 后端：空槽清零。python/sglang/srt/lora/backend/chunked_backend.py 的 init_prefill_cuda_graph_batch_info 同样支持 max_num_requests；prepare_lora_batch 在 prefill 图模式下把 req_seg_indptr[bs + 1 :] 填充为末段边界值、req_weight_indices[bs:] 清零，保证捕获的 MoE kernels 读到稳定的空段。

5. 能力判定与初始化。python/sglang/srt/lora/lora_manager.py 的

supports_prefill_cuda_graph 从 "MoE LoRA 一律不支持" 放宽为：MoE LoRA 在 breakable 或 full 后端下可用，DP attention 仍不支持。

init_prefill_cuda_graph_batch_info 在初始化后端元数据后遍历 base_model.modules()，找到第一个 FusedMoEWithLoRA 层并以 max_bs = max_num_tokens 分配 prefill MoE scratch。python/sglang/srt/lora/layers.py 的 _get_lora_info 同步根据 batch_info 是否为 prefill 快照选择 prefill_moe_cg_buffers。

6. 测试配套。新增 test/registered/e2e/lora/test_lora_moe_prefill_cuda_graph.py：64 个混合 adapter/base 请求，覆盖 Triton/CSGMV × breakable/full 五种组合，比较 eager 与 graph 的 prompt logprobs 与首个 token 分数（阈值 1.0），并用 Prometheus slang:cuda_graph_passes_total 断言两次 prefill 都走了图。扩展

test/registered/lora/test_moe_lora_info.py：

test_moe_graph_metadata_uses_matching_static_buffers 验证 prefill/decode 分别使用各自的静态缓冲区（data_ptr 断言）；TestDenseLoRAPrefillGraph 用 CUDAGraph 实际 capture + replay 三种 ragged 分布，验证 replay 后 adapter 仍保留。

关键文件：

- python/sglang/srt/lora/backend/base_backend.py（模块 LoRA 基类；类别 source；类型 core-logic；符号 init_prefill_cuda_graph_batch_info, init_cuda_graph_moe_buffers, _add_moe_lora_info, _compute_moe_lora_info）：核心改动所在：新增 prefill_moe_cg_buffers 静态缓冲区、init_cuda_graph_moe_buffers 增加 prefill 参数并精简缓冲区集合、_add_moe_lora_info 在 prefill 时使用全部请求槽、_compute_moe_lora_info 增加尾部清零。所有 LoRA 后端的 prefill 图行为都以这里为契约。
- python/sglang/srt/lora/backend/triton_backend.py（模块 Triton 后端；类别 source；类型 core-logic；符号 init_prefill_cuda_graph_batch_info, prepare_lora_batch,

run_lora_a_embedding, compute_sgemm_routing) : 移除 32 请求硬编码上限, 槽位改为按 max_num_requests 或 max_num_tokens 分配; 新增 16-token tile 的 sgemm 网格压缩 (prefill_cuda_graph_sgemm_batch_info) , 并接入 merge_and_chunk_segments 的预填路由。

- python/sglang/srt/lora/lora_manager.py (模块 LoRA 管理; 类别 source; 类型 core-logic; 符号 init_prefill_cuda_graph_batch_info, supports_prefill_cuda_graph, can_use_prefill_cuda_graph) : 能力判定从 "MoE LoRA 一律不支持 prefill 图" 放宽为 breakable/full 下可用; init_prefill_cuda_graph_batch_info 统一初始化后端元数据与 MoE prefill scratch。
- python/sglang/srt/model_executor/runner/prefill_cuda_graph_runner.py (模块 预填图 执行; 类别 source; 类型 data-contract) : 变更入口: 向 LoRA 初始化传入 max_num_requests, 删除 Full 请求槽钳制逻辑; run_once 用 model_capture_mode() 确保 base-only 捕获批次也记录 MoE LoRA kernels。
- python/sglang/srt/lora/backend/chunked_backend.py (模块 CSGMV 后端; 类别 source ; 类型 core-logic; 符号 init_prefill_cuda_graph_batch_info, prepare_lora_batch) : CSGMV 后端同步支持 max_num_requests, 并在 prepare_lora_batch 中对 prefill 图模式清空尾部请求槽元数据, 保证捕获的 MoE kernels 读到稳定空段。
- test/registered/e2e/lora/test_lora_moe_prefill_cuda_graph.py (模块 端到端测试; 类别 test; 类型 test-coverage; 符号 TestMoELoRAPrefillCudaGraph, test_prefill_graph_matches_eager) : 新增端到端回归: 64 个混合 adapter/base 请求 × Triton/CSGMV × breakable/full, 比较 prompt logprobs 与首 token 分数, 并用 Prometheus 指标断言两次 prefill 都走了图。
- test/registered/lora/test_moe_lora_info.py (模块 单元测试; 类别 test; 类型 test-coverage; 符号 test_moe_graph_metadata_uses_matching_static_buffers, TestDenseLoRAPrefillGraph, test_replay_preserves_ragged_adapters) : 新增两处关键单测: 静态缓冲区 data_ptr 一致性校验 (防止 prefill/decode 共用 scratch) , 以及真实 CUDAGraph capture + replay 下 ragged 请求的 adapter 保留。
- python/sglang/srt/lora/layers.py (模块 LoRA 层; 类别 source; 类型 core-logic; 符号 _get_lora_info) : _get_lora_info 根据 batch_info 是否为 prefill 快照选择 prefill_moe_cg_buffers, 保证 MoE LoRA 层在 prefill 图下读到正确的静态缓冲区。
- python/sglang/srt/model_executor/model_runner_components/cuda_graph_setup.py (模块 图配置; 类别 source; 类型 configuration) : 仅更新预填图降级日志文案, 反映 MoE LoRA 在 full/breakable 下已受支持。

关键符号: init_prefill_cuda_graph_batch_info, init_cuda_graph_moe_buffers, _add_moe_lora_info, _compute_moe_lora_info, prepare_lora_batch, supports_prefill_cuda_graph, can_use_prefill_cuda_graph, merge_and_chunk_segments

关键源码片段

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

核心改动所在: 新增 prefill_moe_cg_buffers 静态缓冲区、init_cuda_graph_moe_buffers 增加 prefill 参数并精简缓冲区集合、_add_moe_lora_info 在 prefill 时使用全部请求槽、

`_compute_moe_lora_info` 增加尾部清零。所有 LoRA 后端的 prefill 图行为都以这里为契约。

```
def _add_moe_lora_info(
    self, forward_batch: ForwardBatch, batch_info: LoRABatchInfo
) -> LoRABatchInfo:
    if not self.is_moe_lora:
        return batch_info

    # prefill CUDA graph 与 decode CUDA graph 各用一套静态缓冲区:
    # 捕获时指针被固化进图里, 两套图的大小不同 (decode 按 bs、prefill 按 token 桶),
    # 共用同一块 scratch 会互相踩踏, 因此必须分离。
    prefill = batch_info.is self.prefill_cuda_graph_batch_info
    if batch_info.use_cuda_graph:
        buffers = self.prefill_moe_cg_buffers if prefill else self.moe_cg_buffers
        if prefill and buffers is None:
            raise RuntimeError(
                "prefill MoE-LoRA CUDA graph buffers were not initialized"
            )
        adapter_enabled = buffers["adapter_enabled"]
        token_lora_mapping = buffers["token_lora_mapping"]
    else:
        adapter_enabled = None
        token_lora_mapping = None

    num_tokens, max_len = get_batch_token_counts(forward_batch)

    # 捕获把 align kernel 的请求段数固定死了, 所以 prefill 必须把全部请求槽
    # (包括空槽) 都纳入 segment 视图; 未使用的槽位保持 seg_lens == 0, 作为空段。
    if (
        batch_info.req_seg_indptr is not None
        or batch_info.req_weight_indices is not None
    ):
        assert batch_info.req_seg_indptr is not None
        assert batch_info.req_weight_indices is not None
        num_moe_segments = (
            batch_info.req_weight_indices.shape[0] if prefill else batch_info.bs
        )
        seg_indptr = batch_info.req_seg_indptr[: num_moe_segments + 1]
        req_to_lora = batch_info.req_weight_indices[: num_moe_segments]
    else:
        num_moe_segments = (
            batch_info.weight_indices.shape[0]
            if prefill
            else batch_info.num_segments
        )
        seg_indptr = batch_info.req_seg_indptr[: num_moe_segments + 1]
        req_to_lora = batch_info.weight_indices[: num_moe_segments]

    adapter_enabled, token_lora_mapping = _compute_moe_lora_info(
```

```

        num_tokens,
        seg_indptr,
        batch_info.lora_ranks,
        req_to_lora,
        adapter_enabled,
        token_lora_mapping,
        max_len=max_len,
    )

    batch_info.moe_lora_info = MoELoRABatchInfo(
        seg_indptr=seg_indptr,
        req_to_lora=req_to_lora,
        adapter_enabled=adapter_enabled,
        token_lora_mapping=token_lora_mapping,
    )

    return batch_info

```

评论区精华

核心讨论围绕[prefill_cuda_graph_runner.py](#)第446行新增的[max_num_requests](#)参数展开。

Fridge003 提问: "Why adding the max_num_requests argument, is it safe?" (为什么新增这个参数, 安全吗?)。jybsuper 回应: 该参数用于移除硬编码的 32 请求 LoRA prefill 图上限, 但图捕获仍需要固定大小的元数据——每个请求对应一条 adapter 条目, 因此请求数是一个安全上界; 超出请求数或 token 数上限的批次会回退 eager。Fridge003 随后批准了该 PR。讨论中没有未解决的疑虑, 但参数安全性论证依赖 [can_use_prefill_cuda_graph](#) 与 [prepare_lora_batch](#) 共享同一套 max_bs/max_tokens 判定, 这一点值得后续关注。

- max_num_requests 参数的安全性 (correctness): jybsuper 回应: 该参数用于移除硬编码的 32 请求上限, 但元数据仍需固定大小——每个请求对应一条 adapter 条目, 因此请求数是一个安全上界; 超出请求数或 token 数上限的批次会回退 eager。Fridge003 随后 APPROVED。

风险与影响

- 风险:
 1. MoE 缓冲区契约变化: init_cuda_graph_moe_buffers 重构后, moe_cg_buffers 不再包含 intermediate_cache1/2/3、out_hidden_states、weight_indices_long 等字段, 影响 decode 侧 MoE CUDA graph 路径; 若仍有内核或层代码依赖这些 key, 会在运行时 KeyError。PR 测试全部通过, 大概率是死代码清理, 但风险仍在。
 2. 元数据容量与回退: init_prefill_cuda_graph_batch_info 签名变更 (新增 max_num_requests) 要求所有 LoRA 后端子类同步实现; can_use_prefill_cuda_graph 若与 prepare_lora_batch 的容量判定不一致, 超出容量的批次可能带着 stale 元数据进入图, 导致静默错误。PR 保留了捕获阶段的 assert 兜底。
 3. Full 后端捕获模式: model_capture_mode() 只在 full + _capture_lora 时启用, base-only 批次也会记录 LoRA kernels; 该上下文与现有注意力捕获逻辑 (如 DSV4 的

on_after_cuda_graph_warmup) 的交互未被测试覆盖，而 BCG 路径此前对 post_warmup_hook 有特殊抑制，full 路径无此保护。

4. CSGMV 尾部填充: chunked_backend.py 中 req_seg_indptr[bs + 1 :].fill_(int(req_seg_indptr_cpu[-1])) 依赖 CPU 端末段边界值已就绪，若 bs 与真实段数不一致，空槽会被填充错误边界，使 MoE align kernel 统计出错误的空段长度。
5. 性能收益分布: CSGMV 长输入 (I4096) 提升 22%-23%，但 B64/I128 下仅 18.9%；Triton 在 I4096 下几乎无变化 (-0.39% 到 +0.26%)，说明收益集中在 token 数与请求数差距大的场景，短输入场景不应预期显著加速。
 - 影响: 用户侧: MoE 模型的 LoRA 服务 (如 Qwen3.5-35B-A3B、Inkling NVFP4) 在 prefill 阶段从 eager 切换到 CUDA graph, TTFT 显著下降 (Inkling batch 1 最高提升 225%)；超过 32 个并发请求的 LoRA prefill 批次不再回退 eager。
 - 系统侧: LoRA 与 prefill CUDA graph 的组合覆盖范围从 dense 扩展到 MoE；Full 后端默认配置不变，但 prefill 元数据槽位从固定 32 变为按请求数分配，Breakable 模式下内存占用由 token 桶大小决定。
 - 团队侧: 建立了 prefill 图静态元数据与 scratch 分离的模式，为后续接入更多 LoRA 后端提供模板；测试体系新增了 e2e 图 -eager 一致性验证与 Prometheus 指标断言。
 - 风险标记: MoE 缓冲区契约变化，元数据容量判定一致性，Full 捕获模式交互，跨后端签名同步，收益集中在长输入场景

关联脉络

- PR #38993 [Refactor] Generalize attention graph variants in the decode runner: 同属 CUDA graph 架构演进线: 该 PR 抽取注意力图变体契约并解耦 decode 图运行器，与本 PR 的 prefill 图 LoRA 支持共同推进图执行覆盖范围。
- PR #38585 [AMD][CI] Skip failing Wave test and relax multi-LoRA output check: 同一 LoRA 测试体系: 该 PR 放宽了多 LoRA 输出的容差并维护 AMD CI 稳定性，与本 PR 新增的 LoRA prefill 图 e2e 测试同属 LoRA 质量保障线。