PR #34337 完整报告

sgl-project/sglang

[Spec][LoRA] Support multi-adapter LoRA with EAGLE/NEXTN/DFLASH/DSPARK speculative decoding

合并时间：2026-08-22 05:21

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34337

执行摘要

- 一句话：多适配器 LoRA 支持 EAGLE/NEXTN/DFLASH/DSPARK 投机解码
- 推荐动作：建议精读。这是投机解码与 LoRA 两个核心子系统交叉处的重要功能 PR，设计决策值得学习：共享草稿 + 无损验证的取舍、基于 is_draft_worker 的门控而非配置、cuda-graph 宽度断言防静默错贴、共享模块在所有获取点统一解包，以及“只告警不拒绝 embedding 适配器”的务实选择。测试方法论（以 spec-on vs spec-off 为 oracle、先测复现性地板、用 adapter 身份前缀直接检验 batch 内适配器归属）也很有借鉴价值。后续值得跟进 TBO 与 overlap-loading 两个已知 LoRA 问题的独立修复。

功能与动机

Issue #12903 只启用了 LoRA 与 NGRAM 投机解码的组合，本 PR 将其扩展为 EAGLE / NEXTN / EAGLE3、DFLASH 和 DSPARK 多适配器共批，完成 #11762 中 LoRA 事项的一项。PR body 明确指出旧方案 #28395 依赖 "strips LoRA from a per-draft ServerArgs copy" 的机制，而该机制在 per-runner 值改为构造参数后已不存在，因此需要重新设计："Adapters apply to the target model only; one shared draft runs unadapted. Speculation stays lossless per adapter (verify samples from the adapted target), so only the accept rate is affected."

实现拆解

1. 入口与参数校验：python/sglang/srt/server_args.py 将原先 "LoRA 只兼容 NGRAM" 的硬拒绝替换为 _check_lora_speculative_compatibility()，新增 _LORA_SPEC_ALGORITHMS 白名单（EAGLE / EAGLE3 / DFLASH / DSPARK），并对 --speculative-adaptive、experimental_sgl_trtllm MoE runner、SGLANG_ENABLE_OVERLAP_PLAN_STREAM、DSPARK 非 static ragged 模式逐一给出带原因的拒绝；NEXTN 预先折叠为 EAGLE，因此无需特判。
2. 草稿 runner 排除 LoRA 与共享模块解包：model_runner.py 的 maybe_init_lora_manager 以 is_draft_worker 为门控键；ModelRunner.lora_manager 属性恒存在（未启用时为 None），下游所有 LoRA 路径从“读配置”改为“读 lora_manager is not None”，使草稿 runner 天然跳过。lora/layers.py 新增 unwrap_lora_layer，MTP 草稿（set_lm_head_from_target）、DFLASH worker 和 DSPARK worker 在获取目标模型的 lm_head / embed_tokens 时统一解包，避免草稿把目标适配器 delta 作用于草稿形状

的激活；DSPARK 还顺带修复了 hasattr(lm_head, "weight") 在解包前执行导致误报的 bug。

3. TARGET_VERIFY token 计数统一与宽度断言：lora/utils.py 新增 get_batch_token_counts，集中处理 decode / target-verify / extend 三种模式的（总 token 数、每请求最大 token 数）；base_backend.py 的 _add_moe_lora_info、triton_backend.py 的 prepare_lora_batch、chunked_backend.py 的 _determine_chunk_size 三个调用点全部改走该函数，消除 max(None) / sum(None) 崩溃。triton_backend.py 在 use_cuda_graph 且 TARGET_VERIFY 时断言 spec_info.draft_token_num == batch_info.max_len，防止把按 capture 宽度预填的 seg_lens 静默套用到其他宽度批次（适配器错贴到错误 token 行）。
4. MoE 缓冲区与加载侧告警：lora_manager.py 的 init_lora_cuda_graph_moe_buffers 由 max_bs 改为 max_bs * speculative_num_draft_tokens 计大小，因为 verify capture 喂入的是 bs * draft_token_num 个 token，该 under-size 对 NGRAM + MoE LoRA 同样存在；load_lora_weights / load_lora_weights_from_tensors 加载后调用 warn_if_adapter_targets_embeddings，对 EAGLE 系列下携带 embedding 权重的适配器给出接受率提示（不拒绝，因为会排除所有 MTP 能力基座）。
5. 测试与配套：新增 CPU 单测 test_draft_runner_skips_lora.py、GPU 单测 test_lora_spec_verify_batch_info.py（覆盖 eager 均匀段、graph 宽度断言、MoE token 计数）、E2E test_lora_spec_decoding.py（只断言服务属性而非输出文本，避免贪心解码跨 batch 形状不可复现导致的 flaky），以及两组手动验证工具 test/manual/lora/run_spec_lora_matrix.py（spec-on vs spec-off 逐适配器矩阵）和 check_spec_baseline_divergence.py（判断分歧是否与 LoRA 无关）。

关键文件：

- python/sglang/srt/lora/utils.py（模块 LoRA 工具；类别 source；类型 core-logic；符号 get_batch_token_counts, warn_if_adapter_targets_embeddings）：新增 get_batch_token_counts 统一三种 forward mode 的 token 计数，是 TARGET_VERIFY 段计算修复的核心；新增 warn_if_adapter_targets_embeddings 加载侧告警。
- python/sglang/srt/server_args.py（模块 参数校验；类别 source；类型 core-logic；符号 _check_lora_speculative_compatibility, _LORA_SPEC_ALGORITHMS）：用 _check_lora_speculative_compatibility 替换 NGRAM-only 硬拒绝，是功能开关的入口；白名单与拒绝项设计直接决定支持边界。
- python/sglang/srt/lora/backend/triton_backend.py（模块 LoRA 后端；类别 source；类型 core-logic；符号 prepare_lora_batch）：cuda-graph 宽度断言与 eager 段计算改造所在，直接影响适配器是否被正确贴到 token 行上，是静默错误防护的关键点。
- python/sglang/srt/lora/backend/base_backend.py（模块 LoRA 后端；类别 source；类型 core-logic；符号 _add_moe_lora_info）：MoE LoRA _add_moe_lora_info 的 token 计数改走 get_batch_token_counts，修复 verify 模式下 sum(None) 崩溃。
- python/sglang/srt/lora/lora_manager.py（模块 LoRA 管理；类别 source；类型 core-logic；符号 init_lora_cuda_graph_moe_buffers, load_lora_weights, load_lora_weights_from_tensors）：MoE cuda-graph 缓冲区由按请求计大小改为按 token 计大小（max_bs * draft_token_num），并接入 embedding 告警；同时记录 spec

配置供告警判断。

- python/sglang/srt/model_executor/model_runner.py（模块 模型执行器；类别 source；类型 core-logic；符号 maybe_init_lora_manager）：maybe_init_lora_manager 以 is_draft_worker 为门控键，这是“草稿不加载适配器”设计的落点；lora_manager 属性恒存在供下游判断。
- python/sglang/srt/lora/layers.py（模块 LoRA 层；类别 source；类型 core-logic；符号 unwrap_lora_layer）：新增 unwrap_lora_layer，所有共享模块获取点（MTP lm_head、DFLASH/DSPARK embedding）统一解包，是正确性修复的基础设施。
- python/sglang/srt/speculative/dflash_worker_v2.py（模块 草稿执行器；类别 source；类型 bugfix；符号 forward_batch_generation）：Review 焦点文件：草稿的 embed_module 需解包 LoRA，否则会对草盾形状输入应用目标适配器 delta；a9a6a32235 修复。
- python/sglang/srt/speculative/dspark_components/dspark_worker_v2.py（模块 草稿执行器；类别 source；类型 bugfix；符号 init）：Review 焦点文件：除 lm_head 外，embed_tokens 也要解包；并修复 hasattr(lm_head, "weight") 在解包前执行导致误报的 bug。
- test/registered/lora/test_lora_spec_decoding.py（模块 E2E 测试；类别 test；类型 test-coverage；符号 TestEagle3MultiLoRA, test_adapters_are_applied_under_speculation, test_mixed_adapter_and_wide_batches_are_served, test_speculation_is_active）：新增 E2E 测试：验证多适配器 + EAGLE3 服务可启动、适配器真实生效、混合适配器与超图宽批次可服务、投机确实运行，是 CI 层面对该功能的主要守卫。
- test/registered/unit/lora/test_lora_spec_verify_batch_info.py（模块 单元测试；类别 test；类型 test-coverage；符号 TestLoRASpecVerifyBatchInfo, _verify_batch）：针对 TARGET_VERIFY 段数学的单元测试：eager 均匀段、graph 宽度匹配与拒绝、MoE token 计数，直接对应本次最核心的三处修复。
- test/manual/lora/run_spec_lora_matrix.py（模块 验证工具；类别 test；类型 test-coverage；符号 materialize_adapter, check_adapter_supported, write_adapter_variant, launch）：手动矩阵验证工具：以 spec-on vs spec-off 逐适配器等价为 oracle，覆盖 solo / 混合 / 超宽批次与多种算法，是 lossless 正确性的最强证据来源。
- python/sglang/srt/speculative/eagle_worker_v2.py（模块 草稿执行器；类别 source；类型 bugfix）：MTP 草稿共享目标 lm_head 模块时需要解包 LoRA，是共享 lm_head 路径正确性修复的一部分。

关键符号：get_batch_token_counts, warn_if_adapter_targets_embeddings, _check_lora_speculative_compatibility, unwrap_lora_layer, maybe_init_lora_manager, init_lora_cuda_graph_moe_buffers, prepare_lora_batch, _add_moe_lora_info, set_lm_head_from_target

关键源码片段

python/sglang/srt/lora/utils.py

新增 get_batch_token_counts 统一三种 forward mode 的 token 计数，是 TARGET_VERIFY 段计算修复的核心；新增 warn_if_adapter_targets_embeddings 加载侧告警。

```
"""TARGET_VERIFY 模式下的 token 计数与 embedding 告警（LoRA + 投机解码核心工具）。

TARGET_VERIFY 的 ForwardBatch 走 decode 风格的位置分支，extend_seq_lens 保持
None，但每个请求实际携带 spec_info.draft_token_num 个 token —— 这就是此前
triton eager 路径 max(None)、MoE _add_moe_lora_info sum(None) 崩溃的根源。
"""


def get_batch_token_counts(forward_batch: ForwardBatch) -> Tuple[int, int]:
    """返回 (总 token 数, 每请求最大 token 数)，供 LoRA 段计算使用。"""
    mode = forward_batch.forward_mode
    if mode.is_decode():
        # decode 模式每请求 1 个 token
        return forward_batch.batch_size, 1
    if mode.is_target_verify():
        # 投机验证：每个请求固定 draft_token_num 个 token，统一宽度
        num_tokens_per_req = forward_batch.spec_info.draft_token_num
        return forward_batch.batch_size * num_tokens_per_req, num_tokens_per_req
    if mode.is_extend():
        # 常规 extend：从 CPU 侧计数取最大长度，避免 D2H 拷贝
        return forward_batch.extend_num_tokens, max(forward_batch.extend_seq_lens_cpu)
    raise ValueError(f"Unsupported forward mode: {mode}")


def warn_if_adapter_targets_embeddings(
    lora_name: str,
    embedding_layer_names: Iterable[str],
    speculative_algorithm: Optional[str],
) -> None:
    """EAGLE 系列投机解码下，适配器携带 embedding 权重时给出告警。

    共享草稿模型消费的是 base embedding 权重，适配器对 embedding 的 delta
    不会影响草稿，只会降低接受率；输出不受影响，因此只告警不拒绝。
    """
    if speculative_algorithm not in ("EAGLE", "EAGLE3"):
        return
    modules = sorted(embedding_layer_names)
    if not modules:
        return
    logger.warning(
        "LoRA adapter '%s' targets embedding modules (%s) while EAGLE-family "
        "speculative decoding is enabled. The shared draft consumes their "
        "base weights, so those deltas do not influence drafting and may "
        "reduce the accept rate. Outputs are unaffected.",
        lora_name,
```

```
            ", ".join(modules),
        )
```

python/sglang/srt/server_args.py

用 _check_lora_speculative_compatibility 替换 NGRAM-only 硬拒绝，是功能开关的入口；白名单与拒绝项设计直接决定支持边界。

```
# 各投机算法在 TARGET_VERIFY 阶段呈现 " 每请求固定 token 宽度 ",
# 这是 LoRA 段布局（seg_lens 均匀）成立的前提。
_LORA_SPEC_ALGORITHMS = ("EAGLE", "EAGLE3", "DFLASH", "DSPARK")


def _check_lora_speculative_compatibility(self):
    """校验 LoRA + 投机解码的参数组合是否合法。"""
    if self.speculative_algorithm in ["NGRAM", None]:
        return

    if self.speculative_algorithm not in _LORA_SPEC_ALGORITHMS:
        promoted = (
            " (NEXTN/EAGLE with a Gemma4 assistant draft is automatically "
            "promoted to FROZEN_KV_MTP, which does not support LoRA)"
            if self.speculative_algorithm == "FROZEN_KV_MTP"
            else ""
        )
        raise ValueError(
            "LoRA is only compatible with NGRAM, EAGLE, NEXTN, EAGLE3, "
            "DFLASH, or DSPARK speculative decoding, not "
            f"{self.speculative_algorithm}{promoted}."
        )

    ragged_mode = envs.SGLANG_RAGGED_VERIFY_MODE.get()

    # 逐项列出不支持的组合，原因统一拼接到共享前缀后，
    # 让报错信息能点出具体组合而非只报某个 flag。
    unsupported = [
        (
            self.speculative_algorithm == "DSPARK" and ragged_mode != "static",
            f"does not support SGLANG_RAGGED_VERIFY_MODE={ragged_mode!r}: "
            "the per-request verify lengths it schedules break the "
            "uniform-width LoRA segment layout",
        ),
        (
            self.speculative_adaptive,
            "does not support --speculative-adaptive: the draft is built "
            "from a static ServerArgs snapshot, and the runtime-state "
            "swap does not rebuild LoRA cuda-graph metadata",
        ),
        (
            "experimental_sgl_trtllm"
```

```
            in (self.moe_runner_backend, self.speculative_moe_runner_backend),
            "does not support the experimental_sgl_trtllm MoE runner: its "
            "TopK reads the LoRA config per forward, which the draft "
            "resolves against the target's after its own publish ended",
        ),
        (
            envs.SGLANG_ENABLE_OVERLAP_PLAN_STREAM.get(),
            "does not support SGLANG_ENABLE_OVERLAP_PLAN_STREAM=1: LoRA "
            "batch preparation would run on the plan stream, unordered "
            "against in-flight forwards",
        ),
    ]
    for is_unsupported, reason in unsupported:
        if is_unsupported:
            raise ValueError(
                f"LoRA with EAGLE/NEXTN/EAGLE3 speculative decoding {reason}."
            )
```

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

cuda-graph 宽度断言与 eager 段计算改造所在，直接影响适配器是否被正确贴到 token 行上，是静默错误防护的关键点。

```
  def prepare_lora_batch(self, forward_batch, weight_indices, lora_ranks,
                         scalings, use_cuda_graph):
    bs = forward_batch.batch_size
    if use_cuda_graph:
        assert (
            self.cuda_graph_batch_info is not None
        ), "CUDA Graph batch info is not initialized."
        batch_info = self.cuda_graph_batch_info
        if forward_batch.forward_mode.is_target_verify():
            # seg_lens 是在 capture 时按当时的每请求宽度预填的，之后永不刷新；
            # 若 verify 批次宽度不一致，直接断言失败，而不是把适配器
            # 静默错贴到错误的 token 行上。
            assert forward_batch.spec_info.draft_token_num == batch_info.max_len, (
                "target-verify width "
                f"{forward_batch.spec_info.draft_token_num} does not match "
                f"the captured LoRA cuda-graph width {batch_info.max_len}"
            )
        batch_info.bs = forward_batch.batch_size
        batch_info.num_segments = forward_batch.batch_size
        # ... 后续按 bs 截断 / 清零 seg_lens 并重算 seg_indptr ...
    else:
        # 宽度来自 CPU 侧计数，避免 D2H 传输；
        # generate_sequence_lengths 按 forward mode 生成均匀或变长段。
        _, max_len = get_batch_token_counts(forward_batch)
        seg_lens = generate_sequence_lengths(forward_batch, device=self.device)
        seg_indptr = torch.zeros((bs + 1,), dtype=torch.int32, device=self.device)
        seg_indptr[1:] = torch.cumsum(seg_lens, dim=0)
```

```
# ... 其余 segment / MoE 组装逻辑 ...
```

评论区精华

Review 由 Qiaolin-Yu 提出两个关键问题，均被作者确认是真实缺口并修复：

1. DFLASH 的 *embedding* 模块是否也要解包 LoRA（dflash_worker_v2.py）：作者确认 "it was a real gap"，因为 embed_module 会被用于草稿形状输入（noise_embedding = embed_module(block_ids)），若被 LoRA 包裹，会把目标适配器 delta 套在草稿 token id 上；已在提交 *a9a6a32235* 修复。作者同时说明该配置的适配器只作用于 attention / MLP，unwrap 在该 run 中是 no-op，真正覆盖 wrapped 路径需要 *embedding* 型适配器。
2. DSPARK 草稿路径中 *embedding* 获取点遗漏（dspark_draft.py）：作者确认 "dspark_draft.py *acquires the embedding independently of the worker, so unwrapping only at attach_shared_modules left that path wrapped*"，已一并修复；追查中还发现同文件 hasattr(lm_head, "weight") 检查在解包前执行，BaseLayerWithLoRA 不透传 weight 属性，会导致 LoRA 包裹的 lm_head 被误判为缺少 weight 而报错，解包顺序随之调整。

- DFLASH 草稿的 *embedding* 模块是否需要解包 LoRA (correctness): 已修复：改为 *embed_module* = unwrap_lora_layer(target_model.get_input_embeddings())，并在 1xH200 上验证 DFLASH + 2 适配器仍可服务所有路由（base 2.857、fact 3.333、guard 2.857 *accept length*，输出互相区分）。
- DSPARK 草稿路径中 *embedding* 获取点遗漏与 hasattr 检查顺序 (correctness): 已修复：dspark_draft.py 的 *embed* 获取点同样使用 unwrap_lora_layer，并将 weight 检查移到解包之后；*rebase* 后 attach_shared_modules 移入 *else* 分支，unwrap 只应用一次。

风险与影响

- 风险：核心风险集中在共享草稿设计的固有代价：适配器移动输出分布越远，接受率下降越明显（实验数据：NEXTN 私有适配器 -19~-31%，DFLASH -34%，EAGLE3 -41%），且高并发下 LoRA + *spec* 吞吐可能低于无 *spec*（c=32 时 1419.7 vs 2975.4 tok/s）。triton_backend.py 的宽度断言是 fail-fast 设计，若运行期 draft_token_num 变化会直接报错而非静默降级，属于有意为之但需注意的行为变化。两个已知未修复问题被明确排除在门外：*--enable-two-batch-overlap*（filter_batch 切片 lora_ids 但不重建 LoRA batch info）和 *--enable-lora-overlap-loading*（观察到适配器静默不生效），它们是通用 LoRA 问题而非投机解码问题。兼容性校验的拒绝列表硬编码在 server_args.py，未来新增投机算法时容易遗漏。*embedding* 适配器只告警不拒绝，用户忽略告警可能导致接受率意外下降。此外 *e2e* 贪心解码不可逐 bit 复现，正确性保障依赖手动矩阵工具，CI 覆盖的是“服务属性”而非逐 token 等价。
- 影响：对用户：服务端现在可以直接给 EAGLE / NEXTN / EAGLE3、DFLASH、DSPARK 开启多适配器 LoRA 共批，无需退回 NGRAM 或关闭投机，单并发下 LoRA + *spec* 相比无 *spec* 有 2.12x 提升，但量化结果显示高并发下该组合收益为负。对系统：改动触及 model *executor*、LoRA manager、三个 LoRA 后端和多个 *speculative* worker，TARGET_VERIFY 的 token 计数语义被统一收敛到 *get_batch_token_counts*。对团队：新增的 run_spec_lora_matrix.py 与 *check_spec_baseline_divergence.py* 成为 LoRA + 投

机解码回归验证的专用工具，验证方法论（spec-on vs spec-off、噪声地板测量、adapter 身份前缀检查）值得沉淀为通用实践。

- 风险标记：核心路径变更，TARGET_VERIFY 段布局风险，共享草稿接受率下降，已知 LoRA 问题未修复，e2e 不可逐 bit 复现

关联脉络

- PR #35910 config: publish before the launcher reads effective configuration: 本 PR 的提交 "Read the resolved speculative algorithm from the spec bag" 明确提到 supplied-instance exposure ratchet（该系列引入的测试约束），改为从 config bag 读取解析后的算法名，避免 NEXTN 折叠为 EAGLE 后 embedding 告警被静默跳过。
- PR #35907 config: constructing a config no longer resolves it: 本 PR body 说明取代 #28395 的原因是 per-runner values 已是 constructor arguments、per-draft ServerArgs copy 机制不再存在，与该 config 惰性化系列直接相关。