PR #25681 完整报告

sgl-project/sglang

Add support for generic num_tokens_per_bs in TARGET_VERIFY

合并时间：2026-05-21 02:34

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25681

执行摘要

- 一句话：让 TARGET_VERIFY 的每请求 token 数可自定义
- 推荐动作：该 PR 虽小，但为推测解码的扩展性奠定了基础。建议阅读 spec_registry.py 和 spec_info.py 的新增接口设计，以及 trtllm_mha_backend.py 中元数据计算的统一化。关注后续可能的前向模式重构 PR。

功能与动机

PR 作者在 PR body 中说明：Allow custom spec algorithm to modify num_tokens_per_bs when doing TARGET_VERIFY mode。此外，代码中的 FIXME 注释指出 TARGET_VERIFY 本质上是一种固定长度的 prefill/extend，未来前向模式重构后应移除该接口。

实现拆解

1. 新增可覆写接口：在 CustomSpecAlgo 类（spec_registry.py）和 SpeculativeAlgorithm 枚举（spec_info.py）中添加 supports_target_verify_for_draft() 和 get_num_tokens_per_bs_for_target_verify() 方法。默认实现中，supports_target_verify_for_draft 返回 False（对 CustomSpecAlgo）或仅 DFLASH 返回 True（对 SpeculativeAlgorithm）；get_num_tokens_per_bs_for_target_verify 直接返回 num_draft_tokens，保持向后兼容。
2. 修改 Attention 后端：在 trtllm_mha_backend.py 的 init_forward_metadata_capture_cuda_graph、init_forward_metadata_replay_cuda_graph 和 init_forward_metadata 三个函数中，将原本硬编码 self.speculative_num_draft_tokens 的地方替换为根据 num_tokens // bs 计算出的 tokens_per_req。max_seq_len_q 和 cu_seqlens_q 等也随之动态调整。
3. 修改模型执行器：在 model_runner.py 的 _dummy_run 方法中，不再直接使用 server_args.speculative_num_draft_tokens，而是调用 spec_algorithm.get_num_tokens_per_bs_for_target_verify()。同时，将 draft worker 的检查条件从 is_dflash() 泛化为 supports_target_verify_for_draft()。
4. 修改 CUDA Graph 运行器：在 cuda_graph_runner.py 的初始化中同样调用 get_num_tokens_per_bs_for_target_verify 代替硬编码值。
5. 配套变更：未添加新测试，PR 作者在 checklist 中标记了速度与准确性测试为 N/A。

关键文件：

- python/sglang/srt/speculative/spec_registry.py（模块 推测算法；类别 source；类型 core-logic；符号 supports_target_verify_for_draft, get_num_tokens_per_bs_for_target_verify）：核心变更：在 CustomSpecAlgo 基类中添加 supports_target_verify_for_draft 和 get_num_tokens_per_bs_for_target_verify 可覆写方法，定义扩展点。
- python/sglang/srt/speculative/spec_info.py（模块 推测算法；类别 source；类型 core-logic；符号 get_num_tokens_per_bs_for_target_verify）：在 SpeculativeAlgorithm 枚举中新增相同方法，确保内置算法也能被统一调用。
- python/sglang/srt/layers/attention/trtllm_mha_backend.py（模块 注意力后端；类别 source；类型 core-logic）：所有 TARGET_VERIFY 元数据初始化均改为使用动态计算的 tokens_per_req，移除对 speculative_num_draft_tokens 的硬编码依赖。
- python/sglang/srt/model_executor/model_runner.py（模块 模型执行；类别 source；类型 data-contract）：在 _dummy_run 中改用新接口获取 num_tokens_per_bs，同时泛化 draft worker 合法性检查。
- python/sglang/srt/model_executor/cuda_graph_runner.py（模块 CUDA 图；类别 source；类型 data-contract）：同样改用新接口获取 num_tokens_per_bs，与 model_runner 保持一致。

关键符号：CustomSpecAlgo.supports_target_verify_for_draft, CustomSpecAlgo.get_num_tokens_per_bs_for_target_verify, SpeculativeAlgorithm.get_num_tokens_per_bs_for_target_verify

关键源码片段

python/sglang/srt/speculative/spec_registry.py

核心变更：在 CustomSpecAlgo 基类中添加 supports_target_verify_for_draft 和 get_num_tokens_per_bs_for_target_verify 可覆写方法，定义扩展点。

```
class CustomSpecAlgo:
    # ... 其他方法 ...

    def supports_target_verify_for_draft(self) -> bool:
        """子类可覆盖以标记该算法支持在 draft worker 上运行 target verify"""
        return False

    def get_num_tokens_per_bs_for_target_verify(
        self, num_draft_tokens: int, is_draft_worker: bool
    ) -> int:
        # FIXME: 前向模式重构后应移除。TARGET_VERIFY 本质是固定长度 prefill/extend,
        # 完全支持 CUDA graph。暴露该接口以允许其他用途。
        return num_draft_tokens
```

python/sglang/srt/layers/attention/trtllm_mha_backend.py

所有 TARGET_VERIFY 元数据初始化均改为使用动态计算的 tokens_per_req，移除对 speculative_num_draft_tokens 的硬编码依赖。

```
# init_forward_metadata_capture_cuda_graph 中的 TARGET_VERIFY 分支
elif forward_mode.is_target_verify():
    tokens_per_req = num_tokens // bs # 动态计算 tokens per request
    metadata.cache_seqlens_int32 = self.target_verify_metadata["cache_seqlens"][:bs]
    metadata.cache_seqlens_int32.copy_(seq_lens + tokens_per_req)

    metadata.cu_seqlens_q = torch.arange(
        0, bs * tokens_per_req + 1, tokens_per_req,
        dtype=torch.int32, device=device,
    )
    metadata.max_seq_len_q = tokens_per_req
    metadata.max_seq_len_k = seq_lens.max().item() + tokens_per_req
    # page_table 和 SWA 绑定保持不变

# init_forward_metadata_replay_cuda_graph 中的 TARGET_VERIFY 分支
elif forward_mode.is_target_verify():
    metadata = self.target_verify_metadata[bs]
    metadata.cache_seqlens_int32.copy_(seq_lens + metadata.max_seq_len_q)
    metadata.max_seq_len_k = seq_lens_cpu.max().item() + metadata.max_seq_len_q
    # 移除了原先对 metadata.max_seq_len_q 的重复赋值
```

评论区精华

Review 过程中，merrymercy 在 spec_registry.py 和 spec_info.py 上提出建议，要求在新增方法中添加 FIXME 注释以说明其设计意图与未来重构方向。同时，最初的方法名可能为 get_num_tokens_per_bs，经 review 后改为更具体的 get_num_tokens_per_bs_for_target_verify。所有评论均已解决。

- 为新增方法添加 FIXME 注释 (documentation): 已添加注释并合并。
- 方法命名讨论 (design): 最终代码采用了新名称。

风险与影响

- 风险：
  - 回归风险：在 TARGET_VERIFY 的 CUDA Graph 捕获和回放中，num_tokens_per_bs 的来源从 speculative_num_draft_tokens 变为动态计算。若自定义算法未正确覆写或 num_tokens 分布有变化，可能导致缓存元数据（如 cache_seqlens、page_table）尺寸不匹配，引发错误或性能问题。
  - 性能影响：动态计算 tokens_per_req 增加了除法操作，但可忽略不计；对于标准算法，行为应完全一致，无性能退化。
  - 兼容性：对现有 API 无影响，所有公共接口未改变。
  - 测试覆盖：未添加单元测试，仅依赖现有 CI。建议对自定义算法场景补充测试。
- 影响：
  - 用户影响：使用内置推测算法（EAGLE、DFLASH 等）的用户无感知，行为不变。仅当用户编写自定义 CustomSpecAlgo 子类并覆写新方法时，才体现变更。

- 系统影响：影响 SRT 推理管线中的 TARGET_VERIFY 模式，包括 CUDA Graph 捕获、回放和普通前向元数据初始化。
- 团队影响：为后续前向模式重构（forward mode refactor）提供了基础，该重构将统一 TARGET_VERIFY 与其他固定长度 prefill 的处理。
- 风险标记：核心路径变更，缺少测试覆盖

关联脉络

- 暂无明显关联 PR