PR #26888 完整报告

sgl-project/sglang

[KDA] Add target_verify support for speculative decoding

合并时间：2026-07-25 19:52

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26888

执行摘要

- 一句话：为 KDA 添加 EAGLE 推测解码 target_verify 支持
- 推荐动作：该 PR 值得精读，尤其是 _detect_conv_window_axis 的设计体现了同一代码库支持多种 Conv Layout 的策略。合并后可作为 KDA 推测解码的基础。建议同时阅读相关 PR #30113 和 #28197 了解完整上下文。

功能与动机

GDN (a sister linear attention backend) already supports full EAGLE speculative decoding. This PR achieves feature parity for KDA with specific adaptations, including is_kda=True gating in fused sigmoid recurrent kernel, convolution state transpose, and full convolution weights applied to combined mixed_qkv.

实现拆解

1. KDA 内核 target_verify 入口：在 Triton 内核 fused_sigmoid_gating_delta_rule_update 中添加 target_verify 支持，通过 disable_state_update=True 和 intermediate_states_buffer 参数，使单次前向即可验证多个 draft token 而不修改 SSM 状态。
2. KDAAttnBackend 适配：新增 target_verify 方法委托给 verify_kernel.target_verify；修改 forward_extend，当 forward_mode.is_target_verify() 时跳过 gate 激活（内核内部处理），处理卷积状态转置（(conv_width, qkv_dim) → (qkv_dim, conv_width)）并传递中间状态缓存。
3. 内存池布局感知：新增 _detect_conv_window_axis 函数自动检测卷积窗口轴顺序（优先 GDN 尾轴布局），支持 KDA 的 (K-1, dim) 布局，并通过 conv_window_dedup_enabled(..., is_kda=True) 使 KDA 保持密集布局，避免溢出。
4. 模型 forward 调整：在 KimiLinearForCausalLM.forward 中，当 forward_mode 为 is_target_verify 时也不进行 gate 激活，与 decode 模式一致。
5. 测试覆盖：新增 test_kda_target_verify.py 验证 kernel 等价性（fp32 精确匹配，bf16 误差 <1e-3）；test_ngram_mamba_verify_update.py 测试 commit_mamba_states_after_verify 正确性；test_kda_spec_integration.py 端到端验证正常推理、prefix caching 和 batch 推理无回归。

关键文件：

- python/sglang/srt/mem_cache/memory_pool.py（模块 内存池；类别 source；类型 core-logic；符号 _detect_conv_window_axis）：新增关键函数 _detect_conv_window_axis 以自动检测卷积窗口轴顺序，支持 KDA 的 (K-1, dim) 布局与 GDN 的 (dim, K-1) 布局共存，确保 deduplicated sliding-window view 正确构建。
- python/sglang/srt/layers/attention/linear/kda_backend.py（模块 注意力后端；类别 source；类型 core-logic；符号 target_verify, forward_extend）：核心后端变更：新增 target_verify 方法并将验证委托给 verify_kernel；修改 forward_extend 添加 is_target_verify 分支，处理卷积状态转置和中间 SSM 状态缓存，是 speculative decoding 的关键执行路径。
- python/sglang/srt/models/kimi_linear.py（模块 模型加载；类别 source；类型 data-contract；符号 forward）：模型核心前向函数调整：在 forward 方法中增加 not forward_batch.forward_mode.is_target_verify() 条件，使 TARGET_VERIFY 模式跳过 gate 激活（与 decode 一致），避免重复 gate。
- test/registered/unit/spec/test_ngram_mamba_verify_update.py（模块 状态验证测试；类别 test；类型 test-coverage；符号 TestNgramLastCorrectStepIndices, _compute_last_correct_step_indices, test_linear_chain_all_accepted, test_linear_chain_partial_accept）：新增单元测试，覆盖 commit_mamba_states_after_verify 中 _compute_last_correct_step_indices 的正确性以及 mamba state update 的调用路径。
- test/manual/test_kda_target_verify.py（模块 KDA 内核测试；类别 test；类型 test-coverage；符号 test_kda_target_verify_equivalence, test_kda_target_verify_bf16）：新增 kernel 级等价性测试，严格验证 target_verify 与逐步骤 decode 调用的输出一致性（fp32 精确匹配，bf16 误差 <1e-3），并检查中间状态缓存和原地修改。
- test/manual/test_kda_spec_integration.py（模块 KDA 集成测试；类别 test；类型 test-coverage；符号 test_normal_inference_no_regression, test_prefix_caching_still_works, test_batch_inference, send）：新增端到端手动测试，使用实际 KDA 模型启动服务器，验证无回归、prefix caching 和 batch 推理，确保 speculative 代码整合后不影响基本功能。

关键符号：_detect_conv_window_axis, KDAAttnBackend.target_verify, KDAAttnBackend.forward_extend, KimiLinearForCausalLM.forward

关键源码片段

python/sglang/srt/mem_cache/memory_pool.py

新增关键函数 _detect_conv_window_axis 以自动检测卷积窗口轴顺序，支持 KDA 的 (K-1, dim) 布局与 GDN 的 (dim, K-1) 布局共存，确保 deduplicated sliding-window view 正确构建。

```
def _detect_conv_window_axis(
    self, conv_state_shape: List[Tuple[int, int]], win_len: int
) -> int:
    """
    自动检测卷积窗口轴位置。
    GDN 的 conv_state 形状为 (dim, K-1)，尾轴长度为 K-1；
```

```
    KDA 的形状为 (K-1, dim)，首轴长度为 K-1。
    优先选择 GDN 的尾轴布局，若所有层一致则返回检测到的轴。
    """
    axis = None
    for conv_shape in conv_state_shape:
        # 检查尾轴是否匹配卷积大小
        if conv_shape[-1] == win_len:
            shape_axis = len(conv_shape) - 1 # GDN 布局
        elif conv_shape[0] == win_len:
            shape_axis = 0 # KDA 布局
        else:
            raise ValueError(
                f"conv_state shape {conv_shape} 没有长度为 win_len={win_len} 的轴"
            )
        if axis is None:
            axis = shape_axis
        elif axis != shape_axis:
            raise ValueError(
                f"各层卷积窗口轴不一致: {conv_state_shape}，无法共享 buffer"
            )
    return axis

# 在 _allocate_deduplicated_conv_window 中使用该轴构建物理形状和 as_strided view
```

python/sglang/srt/layers/attention/linear/kda_backend.py

核心后端变更：新增 target_verify 方法并将验证委托给 verify_kernel；修改 forward_extend 添加 is_target_verify 分支，处理卷积状态转置和中间 SSM 状态缓存，是 speculative decoding 的关键执行路径。

```
class KDAAttnBackend(...):

    def target_verify(
        self,
        A_log, dt_bias, q, k, v, a, b,
        *, ssm_states, cache_indices, query_start_loc, **kwargs,
    ) -> torch.Tensor:
        """验证多个 draft token 的 SSM 状态，不修改原始状态。"""
        return self.verify_kernel.target_verify(
            A_log, dt_bias, q, k, v, a, b,
            initial_state_source=ssm_states,
            initial_state_indices=cache_indices,
            cu_seqlens=query_start_loc,
            use_qk_l2norm_in_kernel=True,
            softplus_beta=1.0,
            softplus_threshold=20.0,
            is_kda=True,
            disable_state_update=True,
            intermediate_states_buffer=..., # 外部传入 buffer
            intermediate_state_indices=...,
```

```
            cache_steps=...,
            retrieve_parent_token=None,
        )

    def forward_extend(self, ...):
        # ... 其他逻辑
        if forward_batch.forward_mode.is_target_verify():
            # 跳过 gate 激活，target_verify 内核内部处理
            # KDA 的 conv_state 形状为 (K-1, dim)，需要转置为 (dim, K-1) 以匹配
            # causal_conv1d_update 的期望布局
            conv_state = conv_state.transpose(-2, -1).contiguous()
            output = self.verify_kernel.target_verify(
                ...,
                intermediate_states_buffer=intermediate_ssm_buffer,
                cache_steps=spec_info.draft_token_num,
            )
        else:
            # 正常 extend 路径
            # ...
```

评论区精华

- extra_buffer 支持范围：YAMY1234 指出测试文档中使用了 extra_buffer，但当前分支启动时会断言失败，建议改为 no_buffer。作者已修正文档和测试配置。
- mamba 跟踪开销：YAMY1234 指出 KDAAttnBackend.init_forward_metadata 重写计算了 mamba_track_mask_indices 等，但 KDA 后端不消费，且引入设备同步（.nonzero()），建议移除。作者同意并删除了该重写。
- 卷积窗口轴检测：YAMY1234 指出原 dedup 分配器假定 GDN 的 (dim, K-1) 布局，KDA 使用 (K-1, dim) 导致视图构建错误。作者引入 _detect_conv_window_axis 自动检测轴位置，并通过 is_kda 参数使 KDA 使用密集布局（不参与 dedup）。
- 与 PR #28197 统一：yuan-luo 要求与并行 PR #28197 统一，作者 rebase 后解决了冲突，合并了功能。
  - extra_buffer 支持范围与文档修正 (correctness)：文档已修正，测试配置改为 no_buffer。
  - KDAAttnBackend 中 mamba 跟踪代码开销与可维护性 (performance)：已移除重写函数，留下注释说明。
  - 卷积窗口轴检测与 KDA 布局兼容性 (design)：已实现 _detect_conv_window_axis，KDA 密集布局，GDN 使用 dedup。
  - 与重复 PR #28197 的统一 (other)：已统一为一个版本。

风险与影响

- 风险：
  1. 核心路径变更：修改了 KDAAttnBackend.forward_extend 和 KimiLinearForCausalLM.forward，可能影响非推测解码路径的正常推理，但通过 e2e 测试验证无回归。

2. 功能重叠风险：本 PR 功能与已合并的 PR #30113 存在部分重叠，rebase 时已消除重复的 target_verify 定义和分发逻辑，但可能仍存在隐式依赖。
3. 性能风险：移除了 extend 路径中不必要的 .nonzero() 同步，当前代码无额外同步开销；target_verify 路径批处理 draft token，预期提升 kernel 效率。
4. 测试覆盖不足：kernel 级测试充分，但缺少 CUDA graph 和 radix cache 结合的手动测试（CI 中未覆盖），可能遗漏 rollback 相关 bug。 - 影响：对用户：KimiLinearForCausalLM 现在可以启用推测解码（如 --speculative-algorithm NGRAM），提升推理吞吐。对系统：新增约 700 行代码（测试 570+ 源码 130），改动集中在 KDA 后端和内存池，影响范围有限。对团队：引入 KDA 专用的 conv 布局检测逻辑，需持续维护与 GDN 的差异。 - 风险标记：核心路径变更，功能重叠风险，CUDA graph 测试未覆盖

关联脉络

- PR #28197 [KDA] Add target_verify support for speculative decoding: 并行重复 PR，后通过 rebase 统一，合并了 target_verify 的重复定义和分发逻辑。
- PR #30113 [KDA] Target verify kernel and integration: 该 PR 合并后已包含部分 target_verify 功能，本 PRrebase 后消除了重复的 target_verify 定义和调度逻辑。