PR #28695 完整报告

sgl-project/sglang

[GDN] Support ReplaySSM Ring Spec-Verify

合并时间：2026-07-20 22:06

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/28695

执行摘要

- 一句话：环形缓存替换 GDN 推测验证路径，显存降低 11.5GB
- 推荐动作：该 PR 展示了从学术论文到工程落地的完整过程，包含误差分析、双路径设计、性能调优和验证。对于从事推测解码、线性注意力或显存优化的开发者有很高学习价值。建议仔细阅读 gdn_replayssm_spec_decode.py 中的内核实现和精度讨论。

功能与动机

推测解码中 GDN 验证需要的 intermediate_ssm 缓冲是显存大头，RFC #28511 定义 Part B 的目标是移除每步全状态写入，减少带宽和显存。PR body 中提到 'snapshot buffer itself (intermediate_ssm, 11.48 GB at TP1) is the bulk of spec-decode memory'，同时提到在高 batch 下验证前向受带宽限制，ReplaySSM 正是优化此项。

实现拆解

1. 配置与参数：在 server_args.py 新增 --enable-gdn-replayssm-spec 和对应校验函数 _validate_gdn_replayssm_spec_ring，确保仅在 GDN 模型、线性链（topk<=1）且 Triton 后端时启用。
2. 数据结构和缓存分配：在 memory_pool.py 的 MambaPool.State 和 MambaPool.__init__ 中扩展新的环形缓存张量 (replayssm_d/k/g, rawv/rawk, beta) 以及光标 (write_pos, cache_base, is_flush)。 SpeculativeState.intermediate_ssm 变为可空，启用时不再分配。
3. 验证内核：新增 gdn_replayssm_spec_decode.py，包含两个核心 Triton 内核：gdn_replayssm_spec_circular_kernel（从环形缓存 + 检查点重建输出）和 gdn_replayssm_exact_fold_kernel（在刷新时从原始输入顺序重放，与原始循环更新位一致）。还包含光标管理内核。
4. 集成调用：在 gdn_backend.py::forward_extend 中，当检测到环形缓存就绪时调用 _replayssm_target_verify；在 spec_utils.py::commit_mamba_states_after_verify 中，通过 commit_gdn_replayssm_spec 推进光标而非写 intermediate_ssm；在 hybrid_linear_attn_backend.py 中确保 decode 环形光标前进不干扰 spec 路径。
5. 配套改动：kv_cache_configurator.py 传递 enable_gdn_replayssm_spec 给 MambaPool 构造函数。

关键文件：

- python/sglang/kernels/ops/attention/fla/gdn_replayssm_spec_decode.py（模块 ReplaySSM 内核；类别 source；类型 core-logic；符号 gdn_replayssm_spec_circular_kernel, gdn_replayssm_exact_fold_kernel, _advance_gdn_spec_cursors_kernel, _reset_gdn_replayssm_spec_cursors_kernel）：核心新增文件，包含两个 Triton 内核（圆形缓存重建和闭环精确折叠）以及光标管理函数，是 ReplaySSM spec-verify 的实现主体。
- python/sglang/srt/layers/attention/linear/gdn_backend.py（模块 GDN 后端；类别 source；类型 core-logic；符号 _replayssm_target_verify）：在 forward_extend 中集成 ReplaySSM 验证路径，根据环形缓存就绪状态动态选择调用新内核或回退到原有循环验证。
- python/sglang/srt/server_args.py（模块 配置解析；类别 source；类型 core-logic；符号 _validate_gdn_replayssm_spec_ring）：新增配置项和验证逻辑，确保 ReplaySSM spec-verify 仅在支持条件下启用，包括线性链检查、后端检查和 ring 长度验证。
- python/sglang/srt/mem_cache/memory_pool.py（模块 缓存池；类别 source；类型 core-logic）：扩展 MambaPool 的 State 和 SpeculativeState 数据结构，新增 rawv/rawk/beta 环形缓存和光标，并条件跳过 intermediate_ssm 分配以节省显存。
- python/sglang/srt/speculative/spec_utils.py（模块 推测提交；类别 source；类型 dependency-wiring）：在 commit_mamba_states_after_verify 中添加 ReplaySSM 分支，当环形缓存存在时仅推进光标，不写 intermediate_ssm，同时保留 conv state 的 commit 逻辑。
- python/sglang/srt/layers/attention/hybrid_linear_attn_backend.py（模块 混合注意力；类别 source；类型 core-logic）：修复 decode 环形光标前进逻辑，通过检查 enable_linear_replayssm 标志而非 cursor 张量存在性，避免干扰 spec 环形。
- python/sglang/srt/mem_cache/kv_cache_configurator.py（模块 KV 缓存配置；类别 source；类型 core-logic）：传递 enable_gdn_replayssm_spec 给 MambaPool 构造函数，确保缓存分配正确。

关键符号：_replayssm_target_verify, gdn_replayssm_spec_circular_kernel, gdn_replayssm_exact_fold_kernel, _advance_gdn_spec_cursors_kernel, _reset_gdn_replayssm_spec_cursors_kernel, _launch_gdn_spec, _launch_gdn_exact_fold, gdn_replayssm_spec_decode, commit_gdn_replayssm_spec, _validate_gdn_replayssm_spec_ring

关键源码片段

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

在 forward_extend 中集成 ReplaySSM 验证路径，根据环形缓存就绪状态动态选择调用新内核或回退到原有循环验证。

```
# python/sglang/srt/layers/attention/linear/gdn_backend.py（片段）
# 在 forward_extend 中，当 is_target_verify 为 True 时，检查 ReplaySSM 环形缓存是否就绪
if is_target_verify:
    mamba_pool = self.req_to_token_pool.mamba_pool
    use_replayssm_spec = (
        mamba_cache_params.replayssm_d is not None
```

```
            and getattr(mamba_pool, "replayssm_cache_base", None) is not None
            and not getattr(mamba_pool, "replayssm_is_kda", False)
        )
        if use_replayssm_spec:
            # ReplaySSM 验证路径：从冻结检查点 + 环形缓存重建输出
            core_attn_out = self._replayssm_target_verify(
                layer=layer, query=query, key=key, value=value,
                a=a, b=b, mamba_pool=mamba_pool, layer_cache=mamba_cache_params,
                cache_indices=cache_indices, query_start_loc=query_start_loc,
                draft_token_num=forward_batch.spec_info.draft_token_num,
            )
        else:
            # 回退到原始循环验证（需要 per-draft 快照）
            # 注：当启用 --enable-gdn-replayssm-spec 时 intermediate_state_cache 为 None，
            # 故此处 assert 确保不会静默失败
            assert intermediate_state_cache is not None, (
                "recurrent target_verify fallback requires intermediate_ssm, "
                "which is not allocated under --enable-gdn-replayssm-spec"
            )
            core_attn_out = self.kernel_dispatcher.target_verify(
                A_log=layer.A_log, dt_bias=layer.dt_bias,
                q=query, k=key, v=value, a=a, b=b,
                ssm_states=ssm_states, cache_indices=cache_indices,
                query_start_loc=query_start_loc,
                intermediate_states_buffer=intermediate_state_cache,
                intermediate_state_indices=intermediate_state_indices,
                cache_steps=forward_batch.spec_info.draft_token_num,
                retrieve_parent_token=retrieve_parent_token,
            )
    else:
        # 非验证路径：正常的 GDN 解码
        ...
```

python/sglang/srt/server_args.py

新增配置项和验证逻辑，确保ReplaySSMspec-verify仅在支持条件下启用，包括线性链检查、后端检查和 ring 长度验证。

```
# python/sglang/srt/server_args.py（片段）
# 新增配置项（在 ServerArgs 类中）
enable_gdn_replayssm_spec: A[
    bool,
    "Enable the ReplaySSM GDN spec-verify kernel (Part B of RFC #28511): "
    "a per-slot circular (d, k, g) ring + periodic flush replacing the "
    "recurrent verify's per-draft full-state snapshots. "
    "GDN only, linear-chain (--speculative-eagle-topk in {None, 1}) only. "
    "Reuses --linear-replayssm-cache-len for the ring length.",
] = False

# 在 __post_init__ 中，处理完 speculative_decoding 后调用
```

```
self._validate_gdn_replayssm_spec_ring()

# 验证函数（主要逻辑）
def _validate_gdn_replayssm_spec_ring(self):
    if not self.enable_gdn_replayssm_spec:
        return
    if self.speculative_eagle_topk not in (None, 1):
        raise ValueError(
            "--enable-gdn-replayssm-spec requires a linear draft chain "
            "(--speculative-eagle-topk in {None, 1})"
        )
    decode = self.linear_attn_decode_backend or self.linear_attn_backend
    if decode != "triton":
        raise ValueError(
            "--enable-gdn-replayssm-spec requires the Triton linear-attn "
            "decode backend"
        )
    if self.enable_mamba_extra_buffer():
        raise ValueError(
            "--enable-gdn-replayssm-spec incompatible with mamba extra_buffer"
        )
    # 确保 ring 长度至少为 2 * 最大 draft 数
    if self.linear_replayssm_cache_len < 2 * max_speculative_num_draft_tokens:
        raise ValueError(...)
```

python/sglang/srt/mem_cache/memory_pool.py

扩展 MambaPool 的 State 和 SpeculativeState 数据结构，新增 rawv/rawk/beta 环形缓存和光标，并条件跳过 intermediate_ssm 分配以节省显存。

```
# python/sglang/srt/mem_cache/memory_pool.py（片段）
@dataclass(frozen=True, kw_only=True)
class State:
    conv: List[torch.Tensor]
    temporal: torch.Tensor
    # ReplaySSM 环形缓存（decode 或 spec 共用底层 d/k/g）
    replayssm_d: Optional[torch.Tensor] = None
    replayssm_k: Optional[torch.Tensor] = None
    replayssm_g: Optional[torch.Tensor] = None
    # ReplaySSM spec-verify 额外字段（仅 --enable-gdn-replayssm-spec）
    # rawv / rawk 存储原始输入（激活 dtype，无损），beta 存储 fp32 值
    replayssm_rawv: Optional[torch.Tensor] = None
    replayssm_rawk: Optional[torch.Tensor] = None
    replayssm_beta: Optional[torch.Tensor] = None

@dataclass(frozen=True, kw_only=True)
class SpeculativeState(State):
    # 当 ReplaySSM spec-verify 启用时，intermediate_ssm 不再需要，设为 None
    intermediate_ssm: Optional[torch.Tensor]
    intermediate_conv_window: List[torch.Tensor]
```

```
# 在 MambaPool.__init__ 中
_replayssm_on = enable_linear_replayssm or enable_gdn_replayssm_spec
if _replayssm_on:
    # 分配底层 ring：d [L, V]，k [L, K]，g [L]（fp32）
    ...
    if enable_gdn_replayssm_spec:
        # 额外分配 rawv [L, V]，rawk [L, K]，beta [L]（fp32）
        # 以及光标张量 cache_base [num_slots]，is_flush [num_slots]
        # 并将 intermediate_ssm 设为 None（释放显存）
        ...
```

评论区精华

> kaixih指出之前版本存在长输出退化（AIME0.933recurrentvs0.767ReplaySSMbf16），应明确 scope 并警告用户。yuan-luo 解释根因并实现闭环精确折叠修复，后续展示准确率持平。 kaixih 建议当启用 ReplaySSM 时跳过 intermediate_ssm 分配以实际节省显存，yuan-luo 随后实现该优化。 kaixih 建议验证 ring length >= 2*draft token 数，yuan-luo 在后续提交中修正验证位置。

- 长输出精度退化及修复方案 (correctness): 通过闭环精确折叠修复精度问题，恢复准确率持平。
- 跳过 intermediate_ssm 分配以节省显存 (performance): 实施成功，显存节省约 11.5 GB per GPU。
- 环形长度验证 (correctness): 实现验证，确保环形长度足够。

风险与影响

- 风险：
  - 仅支持 GDN 模型，KDA 和树状验证（topk>1）自动回退到原循环验证，需要确保回退路径的 intermediate_ssm 存在。
  - 与现有 decode ReplaySSM 环形缓存共享张量和光标，若标志错误可能互相干扰（通过检查 enable_linear_replayssm 标志隔离）。
  - 精度虽已修复，但在极端长输出或 fp16 SSM 状态下仍可能有微小漂移（PR 强制使用 fp32 SSM checkpoint）。
  - 新内核依赖 Triton，在非 NVIDIA GPU 上不适用（自动回退）。
  - 默认关闭，无兼容性问题。
- 影响：
  - 用户：显存节省明显（~11.5 GB per GPU at TP1），吞吐量持平，适合高并发推理；需手动开启 --enable-gdn-replayssm-spec。
  - 系统：不影响现有推理路径，所有改动均有条件启用。
  - 团队：新增 Triton 内核需维护，但与 vLLM 社区参考对齐，降低长期成本。
  - 风险标记：仅 GDN 模型，仅线性链，需要 Triton 后端，历史长输出退化已修复，默认关闭

关联脉络

- PR #28511 [RFC] Porting ReplaySSM to SGLang: faster decode and speculative decoding for hybrid (GDN/KDA) models: 本 PR 是 RFC #28511 的 Part B 实现了 spec-verify 部分，RFC 定义了整体架构和 Part A/B 分割。
- PR #28451 [Perf] GDN ReplaySSM decode (Part A of RFC #28511): Part A 实现了 decode ReplaySSM，本 PR 的 spec-verify 复用其环形缓存结构和写入位置光标。
- PR #27658 [RFC] Compact linear spec cache: PR body 中提及相关的 spec-decode 显存优化，与本 PR 目标一致。