PR #35957 完整报告

sgl-project/sglang

Fix recurrent state loss on decode retraction

合并时间：2026-08-25 00:11

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35957

执行摘要

- 一句话：修复 decode retraction 的 recurrent state 静默丢失
- 推荐动作：值得精读。三个设计决策有借鉴价值：用类属性（cpu_copy_carries_mamba）声明式描述池能力，而非在调用方做 isinstance 特判；用共享谓词（uses_ssm_state）消除两处判定逻辑的分裂，避免 " 一个看 attention、一个看 recurrent state" 的夹缝模型；用轻量 stub 类在纯 CPU 单测中完整覆盖备份 / 恢复双分支，验证成本极低。对排查同类 " 同一签名、不同实现 " 导致的静默错误有直接参考意义。

功能与动机

PR body 指出：一个同时具备 sliding-window attention 与 recurrent state 的被 retract 的 decode 请求，会在他人遗留的 recurrent state 上继续解码。根因是 Req.offload_kv_cache 把 mamba_indices 交给 KV pool，而 SWAKVPool.get_cpu_copy 接受该参数却忽略它，只有 full 与 sliding-window 组件被转移；每请求 slot 在 retraction 时释放并可被其他请求复用，恢复的请求就继续使用该 slot 当前的任意状态。HybridLinearKVPool.get_cpu_copy 会转移 mamba 状态，两个 pool 在同一签名下行为不一致，导致静默丢失。在强制 retraction 下，gsm8k over PD 从 0.315（49.5% 不可解析答案）跌到 0.315，修复后 0.850。启动路径也有对应分裂：resolve_decode_retraction_backup 为 hybrid SWA+SSM 模型选择 host_pool，而 _create_unified_radix_cache 拒绝该组合，导致 decode 角色直接以 ValueError: Host-pool retraction does not support Mamba models 退出。

实现拆解

1. 声明式标记 KV pool 能力：在 python/sglang/srt/mem_cache/memory_pool.py 的 KVCache 基类新增类属性 cpu_copy_carries_mamba = False，并在 HybridLinearKVPool 上覆盖为 True。这明确了 get_cpu_copy/load_cpu_copy 是否自行搬运 recurrent state，避免依赖 isinstance 特判或运行时行为推断。
2. Req 备份 / 恢复路径补齐 mamba 状态：在 python/sglang/srt/managers/schedule_batch.py 新增 _mamba_pool_needing_backup 辅助方法：当 KV pool 不自带 mamba 搬运且请求池是 HybridReqToTokenPool 时，返回 mamba_pool；offload_kv_cache 将 mamba_pool.get_cpu_copy(...) 结果存入 RetractionBackup.mamba_cpu，load_kv_cache 在恢复 KV 前先调 mamba_pool.load_cpu_copy(...) 放回状态。RetractionBackup（mem_cache/common.py）新增 mamba_cpu 字段，默认 None，避免破坏既有构造点。

3. 统一 SSM 模型谓词：在 python/sglang/srt/mem_cache/kv_cache_builder.py 新增 uses_ssm_state(model_config)，集中判断 GDN、Mamba2、radix-cache mamba、Kimi-Linear、Lightning 等配置；resolve_decode_retraction_backup 在计算 supports_host_pool 时先取反该谓词，确保 SSM 模型留在 cpu_tensor 路径；build_kv_cache 内联的 is_hybrid_ssm 计算同步替换为该函数，消除两处判定的漂移。
4. 测试配套：新增 test/registered/unit/mem_cache/test_retraction_mamba_backup.py，用轻量 stub 池覆盖两条关键分支：KV pool 不携带 mamba 时状态随 RetractionBackup.mamba_cpu 完整往返；KV pool 自带搬运时不重复拷贝。测试注册为 CPU CI 用例（est_time=5）。
5. CI 验证：作者通过 /rerun-test 指定覆盖 retraction 备份、retract decode、disaggregation 基础、Kimi-Linear PD、Qwen3-Next e2e 等用例集合，1-gpu-5090 与 8-gpu-b200 工作流均通过。

关键文件：

- python/sglang/srt/managers/schedule_batch.py（模块 请求调度；类别 source；类型 core-logic；符号 _mamba_pool_needing_backup, offload_kv_cache, load_kv_cache）：核心修复点：Req.offload_kv_cache / load_kv_cache 在 KV pool 不自带 mamba 搬运时，通过 _mamba_pool_needing_backup 单独备份与恢复 recurrent state，是消除状态丢失的直接改动。
- python/sglang/srt/mem_cache/kv_cache_builder.py（模块 缓存构建；类别 source；类型 core-logic；符号 uses_ssm_state, resolve_decode_retraction_backup, build_kv_cache）：新增共享谓词 uses_ssm_state，同时修复 resolve_decode_retraction_backup 对混合模型的错误选型（host_pool → cpu_tensor）并替换 build_kv_cache 内联判定，消除两处逻辑的分裂。
- test/registered/unit/mem_cache/test_retraction_mamba_backup.py（模块 缓存备份；类别 test；类型 test-coverage；符号 _MambaPool, _Allocator, _req_and_pool, TestRetractionMambaBackup）：新增单元测试，用 stub producer 精确覆盖两条关键分支：KV pool 不携带 mamba 时状态随备份往返、KV pool 自带搬运时不重复拷贝，是该修复的行为契约。
- python/sglang/srt/mem_cache/memory_pool.py（模块 内存池；类别 source；类型 core-logic；符号 KVCache.cpu_copy_carries_mamba, HybridLinearKVPool.cpu_copy_carries_mamba）：在 KVCache 基类声明 cpu_copy_carries_mamba = False，HybridLinearKVPool 覆盖为 True，是区分各 pool 是否自带 mamba 搬运的声明式契约，支撑 _mamba_pool_needing_backup 的判定。
- python/sglang/srt/mem_cache/common.py（模块 缓存工具；类别 source；类型 core-logic；符号 RetractionBackup.mamba_cpu）：RetractionBackup NamedTuple 新增 mamba_cpu 字段，用于携带 KV pool 未搬运的 recurrent state，是备份数据结构的关键扩展。

关键符号：_mamba_pool_needing_backup, offload_kv_cache, load_kv_cache, uses_ssm_state, resolve_decode_retraction_backup

关键源码片段

python/sglang/srt/managers/schedule_batch.py

核心修复点：Req.offload_kv_cache / load_kv_cache 在 KV pool 不自带 mamba 搬运时，通过 _mamba_pool_needing_backup 单独备份与恢复 recurrent state，是消除状态丢失的直接改动。

```
def _mamba_pool_needing_backup(self, req_to_token_pool, allocator):
    # KV pool 自带 recurrent state 转移时（如 HybridLinearKVPool），
    # 调用方不重复备份，避免同一份状态被拷贝两次。
    if allocator.get_kvcache().cpu_copy_carries_mamba:
        return None
    # 只有 HybridReqToTokenPool 持有 mamba_pool，纯 attention 模型无需处理。
    if not isinstance(req_to_token_pool, HybridReqToTokenPool):
        return None
    return req_to_token_pool.mamba_pool


def offload_kv_cache(self, req_to_token_pool, token_to_kv_pool_allocator):
    token_indices = req_to_token_pool.req_to_token[
        self.req_pool_idx, : self.seqlen - 1
    ]
    # 先判断 mamba 状态是否需要单独搬运：SWAKVPool 的 get_cpu_copy 会静默
    # 忽略 mamba_indices 参数，只搬运 full 与 sliding-window 组件，因此
    # 这类 pool 必须由调用方把 recurrent state 放进 RetractionBackup。
    mamba_pool = self._mamba_pool_needing_backup(
        req_to_token_pool, token_to_kv_pool_allocator
    )
    self.retraction_backup = RetractionBackup(
        cpu_tensors=token_to_kv_pool_allocator.get_cpu_copy(
            token_indices, mamba_indices=self.mamba_pool_idx
        ),
        mamba_cpu=(
            mamba_pool.get_cpu_copy(self.mamba_pool_idx.unsqueeze(0))
            if mamba_pool is not None and self.mamba_pool_idx is not None
            else None
        ),
    )


def load_kv_cache(self, req_to_token_pool, token_to_kv_pool_allocator):
    assert self.retraction_backup is not None
    token_indices = req_to_token_pool.req_to_token[
        self.req_pool_idx, : self.seqlen - 1
    ]
    # 恢复顺序：先放回 recurrent state，再放回 KV 组件。
    mamba_cpu = self.retraction_backup.mamba_cpu
    if mamba_cpu is not None and self.mamba_pool_idx is not None:
        req_to_token_pool.mamba_pool.load_cpu_copy(
            mamba_cpu, self.mamba_pool_idx.unsqueeze(0)
```

```
        )
    token_to_kv_pool_allocator.load_cpu_copy(
        self.retraction_backup.cpu_tensors,
        token_indices,
        mamba_indices=self.mamba_pool_idx,
    )
    self.retraction_backup = None
```

python/sglang/srt/mem_cache/kv_cache_builder.py

新增共享谓词 uses_ssm_state，同时修复 resolve_decode_retraction_backup 对混合模型的错误选型（host_pool → cpu_tensor）并替换 build_kv_cache 内联判定，消除两处逻辑的分裂。

```
def uses_ssm_state(model_config) -> bool:
    """模型除 attention KV 外是否还保留 recurrent/conv 状态。"""
    spec = linear_attn_model_spec(model_config)
    return (
        hybrid_gdn_config(model_config) is not None
        or mamba2_config(model_config) is not None
        or (spec.uses_mamba_radix_cache if spec is not None else False)
        or kimi_linear_config(model_config) is not None
        or hybrid_lightning_config(model_config) is not None
    )



# resolve_decode_retraction_backup 中的 host_pool 判定：
# host_pool 只搬运 full 与 sliding-window 组件，因此带 recurrent state 的
# 模型必须留在 cpu_tensor 路径，否则状态会静默丢失（PR body 数据：
# gsm8k 准确率从 0.850 掉到 0.315，且 49.5% 的答案不可解析）。
supports_host_pool = not uses_ssm_state(
    tp_worker.model_runner.model_config
) and (
    isinstance(kv_cache, MHATokenToKVPool)
    or (isinstance(kv_cache, SWAKVPool) and full_tokens_per_layer > 0)
)
```

test/registered/unit/mem_cache/test_retraction_mamba_backup.py

新增单元测试，用 stub producer 精确覆盖两条关键分支：KV pool 不携带 mamba 时状态随备份往返、KV pool 自带搬运时不重复拷贝，是该修复的行为契约。

```
class TestRetractionMambaBackup(unittest.TestCase):
    def test_state_travels_when_kv_pool_leaves_it_behind(self):
        # 模拟 SWAKVPool 这类接受 mamba_indices 但忽略的 pool：若无人单独
        # 备份 recurrent state，复用 slot 的请求会读到别人残留的状态。
        req, pool = _req_and_pool()
        allocator = _Allocator(carries_mamba=False)

        req.offload_kv_cache(pool, allocator)
        self.assertIs(req.retraction_backup.mamba_cpu, MAMBA_STATE)
```

```
        req.load_kv_cache(pool, allocator)
        self.assertIs(pool.mamba_pool.loaded, MAMBA_STATE)

    def test_state_is_not_copied_twice_when_the_kv_pool_carries_it(self):
        # HybridLinearKVPool 自己会搬运 recurrent state，调用方不应重复备份。
        req, pool = _req_and_pool()
        allocator = _Allocator(carries_mamba=True)

        req.offload_kv_cache(pool, allocator)
        self.assertIsNone(req.retraction_backup.mamba_cpu)

        req.load_kv_cache(pool, allocator)
        self.assertIsNone(pool.mamba_pool.loaded)
```

评论区精华

本 PR 没有 review 评论或 review thread，核心论证全部写在 PR body 中：作者详细推演了根因——SWAKVPool.get_cpu_copy 接受 mamba_indices 却忽略，与 HybridLinearKVPool 在同一签名下行为不一致，属于静默错误；并解释了为何无法做 bit-exact 验证——恢复请求通过 extend 路径重算边界 token，而未中断运行通过 decode 路径，两条路径的 sconv 内核并不 bit 一致，因此改用按物理 slot 对每个转移组件做 checksum 校验。Issue 评论中作者两次触发 /rerun-test，指定覆盖 test_decode_retraction_backup.py、test_retraction_mamba_backup.py、test_retract_decode.py、test_disaggregation_basic.py、test_kimi_linear_pd_dcp4.py、test_qwen3_next_models.py，并在说明中强调：retraction 路径本身、PD decode 侧、HybridLinearKVPool 自带状态的 branch、以及纯 attention 模型都需要覆盖。

- 再跑测试覆盖范围 (testing): 1-gpu-5090 上 2 个测试与 8-gpu-b200 上的 test_kimi_linear_pd_dcp4.py 均通过，修复行为得到验证。

风险与影响

- 风险：
  1. 运行时配置推断依赖：uses_ssm_state 是新增的集中谓词，依赖 model_config 判断，未来若新增 linear-attention/SSM 模型类型而忘记在此登记，会重新落入 host_pool + 状态丢失路径。建议在模型配置注册处增加断言或测试守护。
  2. NamedTuple 字段扩展的序列化兼容性：RetractionBackup 新增 mamba_cpu 字段，所有既有构造点依赖默认值 None 保持兼容，但涉及 msgpack/ 序列化 round-trip 的场景（如跨进程）没有新增专门测试，存在潜在漏传风险。
  3. 验证精度受限：由于 extend/decode 路径的 sconv 内核不 bit 一致，无法做逐位对比，只能依赖 checksum 与端到端指标，理论上存在非 mamba 组件状态损坏但未被 checksum 覆盖的盲区。
  4. host_pool 路径收窄：SSM 模型被强制留在 cpu_tensor 备份路径，host_pool 的显存优化对这些模型不再生效；不过这些模型此前根本无法启动（ValueError），因此这是修复而非回退。

- 影响：
  1. 模型影响：同时具备 sliding-window attention 与 recurrent state 的混合模型（如 Inkling-Small、Qwen3-Next、Kimi-Linear 等）在 PD 分离模式下的 retraction 从 " 无法启动 " 或 " 静默错误解码 " 变为正确恢复；纯 attention 模型不受影响，HybridLinearKVPool 自带的 mamba 搬运路径也不会重复备份。
  2. 系统影响：修复了 decode 角色启动时的 ValueError，使混合模型可正常部署 PD；SSM 模型在 retraction 时多一次 mamba 状态拷贝与恢复，开销集中在 retraction 事件本身，对稳态吞吐影响可忽略。
  3. 团队影响：确立了 cpu_copy_carries_mamba 声明式约定，后续新增 KV pool 类型必须明确表态是否自带状态搬运，避免再次出现同一签名下的静默不一致。 - 风险标记：核心路径变更，静默正确性修复，运行时配置推断依赖，缺少 bit-exact 验证

关联脉络

- PR #35840 Add PD test for inkling with mxfp8 KV: PR body 明确说明 #35840 的配对测试覆盖本修复的配置（Inkling + MXFP8 KV + retraction）；两者改动同一批 retraction/mamba 备份文件（schedule_batch.py、kv_cache_builder.py、memory_pool.py、common.py），属于同一功能线的连续演进。