PR #35689 完整报告

sgl-project/sglang

Skip empty linear-attention state buffers in PD transfer

合并时间：2026-08-21 01:00

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35689

执行摘要

- 一句话：跳过空状态 buffer，修复 Inkling PD 传输失败
- 推荐动作：值得精读。根因分析链路（黑名单遮蔽 → 零长度 region → 退化 temporal shape）是典型的分布式系统疑难排查示例；commit 历史展示了从注册端过滤到生产端过滤的演进，可作为过滤位置选择的设计参考；新增单测用 stub 精确构造退化场景，是低依赖、快速复现问题的良好范例。

功能与动机

PD 分离无法为 Inkling 传输状态，每个请求都以 Failed to send kv chunk 失败，session 随即进入黑名单，后续请求全部报 remote mooncake session ... is not alive，黑名单把第一个真实错误完全掩盖。调高 mooncake 日志后可见根因：Transfer Engine does not support zero length memory region。MambaPool._iter_transfer_state_tensors 只跳过 None 或 _NON_TRANSFER_STATE_FIELDS 字段，但 ShortConv 层没有循环时间状态，temporal 形状退化，pool 为每层分配零字节张量，被当作可传输状态广播；一次失败的 batch 注册还让真实 buffer 未注册，导致失败信息指向别处。实测 Inkling-Small 中 mamba 组件 294 个条目有 42 个空条目，全部来自 temporal。作者强调这不是回归：退化 temporal 形状自 #31681 初始 Inkling 支持起就存在。

实现拆解

1. 根因定位：在 python/sglang/srt/mem_cache/memory_pool.py 的 _iter_transfer_state_tensors 中，遍历 vars(self.mamba_cache) 时仅排除 _NON_TRANSFER_STATE_FIELDS 与 None 字段，退化 temporal 形状产生的空张量仍会被产出并进入 RDMA 注册列表。
2. 修复落地：在迭代器内部对每个 state_tensor 增加 numel() == 0 判断并 continue，从源头排除空 buffer。选择迭代器而不是 transfer 后端，是因为 get_contiguous_buf_infos、get_state_dim_per_tensor 等 5 个调用方都从该迭代器构建并行 per-tensor 列表（指针、长度、条目长度、可切片维度、字段名），源头过滤可保证列表始终对齐，且 mooncake 与 NIXL 两条注册路径、prefill 与 decode 两侧共用同一逻辑、行为保持一致。
3. 测试配套：新增 test/registered/unit/mem_cache/test_mamba_state_transfer_buffers.py，用 object.new(MambaPool) 构造最小 stub（_pool），分别以退化 temporal（shape [2, 3, 0, 0, 0]）和正常 temporal（shape [2, 3, 6, 7, 8]）验证：空 buffer 不被广播且 lens/item_lens 不含 0、temporal 仍被正常广播、get_state_dim_per_tensor 与

get_contiguous_buf_infos 返回列表长度保持对齐。测试注册为 CPU CI（base-a-test-cpu，预计 5 秒）。

4. 验证与部署：单节点 PD（一个 prefill、一个 decode，TP = 2）、mooncake over IB、prefill 侧 hierarchical cache 下，sgl-eval aime26（thinking on，max_tokens = 131072）得到 score = 0.9667、error_rate = 0.00%、stop_rate = 1.00%，约 25k completion tokens/ 题的长生成验证 decode 侧卷积状态在传输数据上持续正确推进。作者还 rerun 了 test_mamba_state_transfer_buffers.py（CPU）与 test_kimi_linear_pd_dcp4.py（8 卡 B200 GPU 集成），均通过。

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 MambaPool, _iter_transfer_state_tensors, get_contiguous_buf_infos, get_state_dim_per_tensor）：核心修复位置：_iter_transfer_state_tensors 新增 numel() == 0 跳过，阻止零字节状态 buffer 进入 RDMA 注册，修复 Inkling 在 PD 分离下所有请求失败的根因。
- test/registered/unit/mem_cache/test_mamba_state_transfer_buffers.py（模块 状态传输；类别 test；类型 test-coverage；符号 _pool, TestMambaStateTransferBuffers, test_conv_only_state_advertises_no_empty_buffer, test_temporal_state_is_still_advertised）：新增单测覆盖三类关键行为：空 buffer 不被广播、temporal 仍被广播、state dims 与 buffer 列表保持对齐，防止回归。

关键符号：_iter_transfer_state_tensors, get_contiguous_buf_infos, get_state_dim_per_tensor, _pool, test_conv_only_state_advertises_no_empty_buffer, test_temporal_state_is_still_advertised, test_dims_stay_aligned_with_buffers

关键源码片段

python/sglang/srt/mem_cache/memory_pool.py

核心修复位置：_iter_transfer_state_tensors 新增 numel() == 0 跳过，阻止零字节状态 buffer 进入 RDMA 注册，修复 Inkling 在 PD 分离下所有请求失败的根因。

```
class MambaPool:
    # 中间计算状态不参与 RDMA 传输，显式排除。
    _NON_TRANSFER_STATE_FIELDS = frozenset({
        "intermediate_ssm",
        "intermediate_conv_window",
        "replayssm_d",
        "replayssm_k",
        "replayssm_g",
        "replayssm_rawv",
        "replayssm_rawk",
        "replayssm_beta",
    })

    def _iter_transfer_state_tensors(self):
        """产出可传输的状态张量及其 per-slot 切片轴。
```

```
        所有 transfer 访问器都从这里取数，因此在此处过滤空张量，
        可以保证各调用方拿到的 per-tensor 列表始终对齐。
        """
        for field, value in vars(self.mamba_cache).items():
            # None 字段与显式排除的中间状态不参与传输。
            if field in self._NON_TRANSFER_STATE_FIELDS or value is None:
                continue
            tensors = value if isinstance(value, list) else [value]
            # conv 状态在第二个 per-slot 轴切片，其他字段在第一个轴。
            slice_axis = self.conv_slice_axis if field == "conv" else 0
            for state_tensor in tensors:
                # ShortConv 层没有循环时间状态，temporal 是退化形状，
                # pool 分配出零字节张量。RDMA 引擎拒绝零长度 region，
                # 且一次失败注册会连累同批次真实 buffer，因此必须跳过。
                if state_tensor.numel() == 0:
                    continue
                yield field, state_tensor, slice_axis

    def get_contiguous_buf_infos(self):
        """返回 RDMA 注册所需的逐层指针、字节数与条目大小。"""
        data_ptrs, data_lens, item_lens = [], [], []
        for _, state_tensor, _ in self._iter_transfer_state_tensors():
            # state_tensor 首个维度是层数，逐层展开为独立注册条目。
            data_ptrs += [state_tensor[i].data_ptr() for i in range(self.num_mamba_layers)]
            data_lens += [state_tensor[i].nbytes for i in range(self.num_mamba_layers)]
            item_lens += [state_tensor[i][0].nbytes for i in range(self.num_mamba_layers)]
        return data_ptrs, data_lens, item_lens

    def get_state_dim_per_tensor(self):
        """返回每个状态张量可切片维度尺寸，供传输规划使用。"""
        dim_per_tensor = []
        for _, state_tensor, slice_axis in self._iter_transfer_state_tensors():
            # state_tensor 形状为 [num_layers, size + 1, sliceable_dim, ...],
            # Kimi conv 状态会把两个 per-slot 轴转置为 [K-1, dim]。
            axis = 2 + slice_axis
            sliceable_dim = state_tensor.shape[axis]
            # 每个层都有独立 data_ptr 条目，因此维度也按层重复。
            dim_per_tensor += [sliceable_dim] * self.num_mamba_layers
        return dim_per_tensor
```

评论区精华

本 PR 没有 review 评论（review_comments_count = 0），核心决策体现在提交历史与 PR body 中：

- 跳过逻辑放在哪里（register 端 vs state producer）：最初的提交是 skip zero length kv regions on register，随后作者提交 move the skip to the state producer 把过滤上移到 _iter_transfer_state_tensors。PR body 给出理由：5 个调用点从它构建并行 per-tensor 列表，源头过滤可保持对齐，且 mooncake 与 NIXL 两条注册路径、prefill 与 decode 两侧

共用同一来源，避免各调用点重复处理与列表错位。

- 黑名单屏蔽真实错误的定位教训：PR body 指出 session blacklisting 让后续请求只报 remote mooncake session ... is not alive，把首个 Transfer Engine does not support zero length memory region 完全隐藏；只有调高 mooncake 日志才能看到注册先失败于空 buffer、真实 buffer 也跟着未注册的真实顺序。
- 跳过逻辑的放置位置：register 端还是 state producer (design): 最终选择在 _iter_transfer_state_tensors 中按 numel()==0 过滤空张量，避免在每个调用点重复处理。
- session blacklisting 屏蔽真实错误，影响根因定位 (other): 根因在 state producer 修复后，黑名单不再被空 buffer 触发；该案例也说明黑名单机制会掩盖根因，值得在日志或错误上报层面改进。

风险与影响

- 风险：行为变更面：所有消费 _iter_transfer_state_tensors 的调用方（mooncake 与 NIXL 注册、prefill 与 decode 侧）都会跳过空张量。当前全部调用方都通过同一迭代器构建并行列表，新增测试 test_dims_stay_aligned_with_buffers 覆盖了维度对齐，但未来若出现按固定索引消费列表的第三方调用方，条目数变化可能造成错位。 空 buffer 语义假设：修复假设 numel() == 0 即无状态可传。若未来某模型用零字节 buffer 承载占位语义（如依赖固定条目数进行字段对齐），该判断会误跳过；但零长度 region 在 RDMA 引擎层面本就不被支持，因此该假设在当前传输架构下安全。 验证覆盖：CPU 单元测试是 stub 级验证，PD 传输正确性依赖真实环境：rerun 的 test_kimi_linear_pd_dcp4.py（8 卡 B200）通过，且 AIME26 长生成评估（每条约 2.5 万 completion tokens）提供强正确性证据；Base CI 门禁未跑出结果，Extra CI 通过，AMD 门禁当时仍在排队。
- 影响：用户侧：Inkling 系列模型在 PD 分离下从每个请求 100% 失败变为可用，AIME26 达到 score = 0.9667、error_rate = 0.00%，且长生成未出现截断或状态错位。 系统侧：RDMA 注册不再提交零长度 region，消除 batch registration 连锁失败；mooncake 与 NIXL 两条路径行为一致，prefill 与 decode 因同一迭代器而保持同步。 团队侧：建立了在 state producer 源头过滤空张量的模式，避免各调用点重复处理；新增 CPU 测试可复用于后续 Mamba 状态传输改动。
- 风险标记：核心路径变更，空 buffer 语义依赖模型，列表对齐需测试保障，PD 传输行为变化

关联脉络

- PR #31681 Initial Inkling support: PR body 明确指出退化 temporal 形状与重构前同样广播 temporal 的行为自 #31681 初始 Inkling 支持起就存在，本 PR 是修正该历史行为而非引入回归。
- PR #35293 test: switch the Inkling-Small NVFP4 deterministic suite to DSPARK: 同属 Inkling-Small 模型功能线，前者补测试矩阵覆盖，本 PR 解锁 PD 传输，共同推进该模型的可靠性建设。