PR #27004 完整报告

sgl-project/sglang

fix(disagg): correct DSA/SWA state-page transfer mismatch in PD disaggregation

合并时间：2026-06-03 14:33

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27004

执行摘要

- 一句话：修复 PD 分离中 DSA/SWA 状态页传输不匹配
- 推荐动作：建议阅读者精读该 PR，了解如何通过边界防御和长度限制修正复杂的分布式传输 bug。group_concurrent_contiguous 的防御性设计思路值得参考。对于 DSA 模型分离部署团队，应尽快合并。

功能与动机

在 PD 分离部署的 DSA (NSA) 模型上，使用 mooncake 传输后端时，预填充端在发送最后一块时，fill_ids 已包含采样 token，导致状态页列表比解码端注册的长度多一页，进而触发 group_concurrent_contiguous 中 NumPy 广播形状不匹配崩溃。详细见 PR body：'On the last chunk, the prefill side enumerates its DSA/SWA state-page list over seq_len = len(req.fill_ids). By send time fill_ids already includes the token sampled during prefill... But the decode side registers its destination state pages over len(origin_input_ids)'。

实现拆解

步骤

1. 防御 group_concurrent_contiguous 函数(python/sglang/srt/disaggregation/common/utils.py)：
   - 当 src_indices 或 dst_indices 任一为空时，立即返回空列表，而不是只在 src 空时返回。
   - 当两者均非空但长度不一时，抛出 ValueError 提供明确错误信息。 这是对之前仅检查 src 为空的补充，避免仅 dst 为空时发生 NumPy 广播错误。
2. 修正预填充状态页序列长度(python/sglang/srt/disaggregation/prefill.py)：
   - 在 send_kv_chunk 方法中，将 seq_len 的计算从 len(req.fill_ids) 改为 min(len(req.fill_ids), len(req.origin_input_ids))。
   - 这保证状态页枚举范围与主池传输范围（已由 end_idx 限制）一致，不会因为采样 token 多出一页。
3. 添加单元测试(test/registered/unit/disaggregation/test_disaggregation_wire.py)：
   - 新增 TestGroupConcurrentContiguous 类，包含 6 个测试：
   - test_single_contiguous_group：正常连续分组

- test_splits_on_discontiguous_indices：非连续分割
- test_both_empty：双方空
- test_empty_src_nonempty_dst：源空目标非空（返回空）
- test_nonempty_src_empty_dst：源非空目标空（回归测试，返回空而非崩溃）
- test_mismatched_nonempty_lengths_raise：长度不等时抛出 ValueError

关键文件：

- python/sglang/srt/disaggregation/common/utils.py（模块 分离传输；类别 source；类型 core-logic；符号 group_concurrent_contiguous）：核心防御逻辑：修复 group_concurrent_contiguous 函数，增加空数组守卫和长度校验，避免 NumPy 广播错误和静默误分组。
- python/sglang/srt/disaggregation/prefill.py（模块 分离传输；类别 source；类型 core-logic；符号 send_kv_chunk）：根因修复：修改 send_kv_chunk 中 seq_len 计算，确保状态页枚举范围与解码端注册长度一致。
- test/registered/unit/disaggregation/test_disaggregation_wire.py（模块 分离测试；类别 test；类型 test-coverage；符号 TestGroupConcurrentContiguous, _arr, test_single_contiguous_group, test_splits_on_discontiguous_indices）：新增单元测试，覆盖 group_concurrent_contiguous 的所有边界场景，确保回归防护。

关键符号：group_concurrent_contiguous, send_kv_chunk

关键源码片段

python/sglang/srt/disaggregation/common/utils.py

核心防御逻辑：修复 group_concurrent_contiguous 函数，增加空数组守卫和长度校验，避免 NumPy 广播错误和静默误分组。

```
def group_concurrent_contiguous(
    src_indices: npt.NDArray[np.int32], dst_indices: npt.NDArray[np.int32]
) -> Tuple[List[npt.NDArray[np.int32]], List[npt.NDArray[np.int32]]]:
    """Vectorised NumPy implementation."""
    # src/dst indices are transferred pairwise, so an empty side means there is
    # nothing to transfer. Guarding both sides (not just src) avoids a cryptic
    # NumPy broadcast error from np.diff() below when only one side is empty, e.g.
    # a non-empty prefill DSA/SWA state list paired with an empty decode registration.
    if src_indices.size == 0 or dst_indices.size == 0:
        return [], []

    if src_indices.size != dst_indices.size:
        raise ValueError(
            "group_concurrent_contiguous requires equal-length src/dst index arrays, "
            f"got {src_indices.size} and {dst_indices.size}"
        )

    brk = np.where((np.diff(src_indices) != 1) | (np.diff(dst_indices) != 1))[0] + 1
    src_groups = np.split(src_indices, brk)
```

```
    dst_groups = np.split(dst_indices, brk)

    src_groups = [g.tolist() for g in src_groups]
    dst_groups = [g.tolist() for g in dst_groups]

    return src_groups, dst_groups
```

python/sglang/srt/disaggregation/prefill.py

根因修复：修改 send_kv_chunk 中 seq_len 计算，确保状态页枚举范围与解码端注册长度一致。

```
        if last_chunk:
            self.disagg_metadata_buffers.set_buf(req)

            # fill_ids includes the token sampled during prefill, but decode
            # registers state pages over origin_input_ids (DecodePreallocQueue)
            # and the main pool send is clamped to end_idx above. Matching that
            # length here avoids emitting an extra state page when the sampled
            # token crosses a page boundary, which mismatched src/dst lengths in
            # group_concurrent_contiguous.
            seq_len = min(len(req.fill_ids), len(req.origin_input_ids))

            def _mamba_payload():
                return [
                    self.req_to_token_pool.req_index_to_mamba_index_mapping[
                        req.req_pool_idx
                    ]
                    .cpu()
                    .numpy()
                ]

            def _swa_payload():
                window_size = self.sliding_window_size
                window_start = max(0, seq_len - window_size)
                window_start = (window_start // page_size) * page_size
                window_kv_indices_full = self.req_to_token_pool.req_to_token[
                    req.req_pool_idx, window_start:seq_len
                ]
                window_kv_indices_swa = (
                    self.token_to_kv_pool_allocator.translate_loc_from_full_to_swa(
                        window_kv_indices_full
                    )
                )
                return kv_to_page_indices(
                    window_kv_indices_swa.cpu().numpy(), page_size
                )

            def _dsa_payload():
                kv_indices_full = self.req_to_token_pool.req_to_token[
```

```
            req.req_pool_idx, :seq_len
        ]
        return kv_to_page_indices(kv_indices_full.cpu().numpy(), page_size)
```

test/registered/unit/disaggregation/test_disaggregation_wire.py

新增单元测试，覆盖 group_concurrent_contiguous 的所有边界场景，确保回归防护。

```
class TestGroupConcurrentContiguous(unittest.TestCase):
    @staticmethod
    def _arr(values):
        return np.array(values, dtype=np.int32)

    def test_single_contiguous_group(self):
        src = self._arr([10, 11, 12])
        dst = self._arr([5, 6, 7])
        self.assertEqual(
            group_concurrent_contiguous(src, dst),
            ([[10, 11, 12]], [[5, 6, 7]]),
        )

    def test_splits_on_discontiguous_indices(self):
        src = self._arr([10, 11, 20])
        dst = self._arr([5, 6, 7])
        self.assertEqual(
            group_concurrent_contiguous(src, dst),
            ([[10, 11], [20]], [[5, 6], [7]]),
        )

    def test_both_empty(self):
        self.assertEqual(
            group_concurrent_contiguous(self._arr([]), self._arr([])), ([], [])
        )

    def test_empty_src_nonempty_dst(self):
        self.assertEqual(
            group_concurrent_contiguous(self._arr([]), self._arr([1, 2])), ([], [])
        )

    def test_nonempty_src_empty_dst(self):
        # Regression: a non-empty source paired with an empty destination must not
        # raise a NumPy broadcast error (observed transferring DSA sparse-attention
        # state on a disaggregated GLM deployment when decode registered zero dst indices).
        self.assertEqual(
            group_concurrent_contiguous(self._arr([1, 2]), self._arr([])), ([], [])
        )

    def test_mismatched_nonempty_lengths_raise(self):
        with self.assertRaises(ValueError):
            group_concurrent_contiguous(self._arr([1, 2, 3]), self._arr([1, 2]))
```

评论区精华

在 review 中，gemini-code-assist[bot] 提出了两个意见：

- 关于空目标数组的返回行为：在 group_concurrent_contiguous 中，当 src_indices 非空但 dst_indices 为空时，PR 实现返回 []，[]。评审认为这会静默丢弃待传输的状态，导致 decode 端状态缺失，建议改为抛出 ValueError。PR 维持了返回空的设计，未采纳该意见。
- 对应的测试用例：评审建议将 test_nonempty_src_empty_dst 测试改为断言抛出错误，PR 同样未修改，保留了当前行为。 最终，项目成员 ShangmingCai 批准了该 PR。设计决策倾向于认为空 dst 意味着没有页面需要注册，返回空是合理的安全规避。
- 空目标数组返回行为的合理性(design):PR维持返回空列表的设计，未采纳抛出异常的建议。
- 相应测试用例应改为断言异常 (testing): PR 保留了原测试，未修改。

风险与影响

- 风险：主要风险：
  - 静默状态遗漏：当 dst_indices 为空而 src_indices 非空时，状态页不会被传输，可能导致 decode 端状态缺失。当前设计假设这种情况不会发生（因为解码端总是注册了相同数量的页面），但若出现不一致，不会收到错误信号。
  - 防御过度：增加了运行时长度校验，但代价很小。
  - 性能影响：仅增加两个大小检查和一次比较，无显著影响。 总体风险较低，但 reviewer 指出的空 dst 情况需关注。
- 影响：影响范围：仅限于使用 DSA/SWA 稀疏注意力的模型（如 GLM-DSA-FP8）且在 PD 分离部署下的用户。修复了预填充转解码时的崩溃，确保状态页面正确传输。无模型向前兼容问题，无性能退化。测试覆盖了核心边界。
- 风险标记：空 dst 静默忽略，长度校验保护，防御编程提升稳健性

关联脉络

- PR #27011 [Bugfix] Clean up failed NIXL sender state: 同属 disaggregation 传输层修复，都涉及状态清理和崩溃修复。