PR #24188 完整报告

sgl-project/sglang

[NIXL][XPU] Use np.uint64 for pointer/length arrays in disaggregation KV transfer

合并时间：2026-05-06 10:09

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24188

执行摘要

- 一句话：XPU 上 NIXL 指针数组改用 np.uint64 防溢出
- 推荐动作：该 PR 是典型的平台适配修正，值得对 XPU 或异构编程开发者阅读。设计上，在传输函数入口统一做 dtype 转换，避免分散在代码各处，是一个好做法。

功能与动机

根据 PR 描述：Intel XPU device addr. pointers can exceed np.int64 range (bit 63 set), causing overflows and incorrect arithmetic when using numpy. 因此需要将数组 dtype 升至 uint64 以保证安全。

实现拆解

1. 在 conn.py 的 _send_kvcache_generic 入口，将 src_data_ptrs、dst_data_ptrs、item_lens 三个列表用 np.array(..., dtype=np.uint64) 转换，并添加注释说明原因。
2. 把预计算 block 起始和长度的 np.fromiter 调用 dtype 从 np.int64 改为 np.uint64。
3. 修改嵌套函数 make_req_array：empty 分支返回 np.empty((0, 3), dtype=np.uint64)；非空分支使用 .astype(np.uint64, copy=False) 确保拼接后的数组类型，并用 np.full_like(..., dtype=np.uint64) 生成 GPU ID 列。
4. 同步修改 send_kvcache_slice 内相似的 make_req_array 实现。
5. 新增 test/registered/disaggregation/test_disaggregation_xpu.py 集成测试，端到端验证 XPU 上 NIXL 传输可用性。
6. 更新 docs/platforms/xpu.md，增加 [Experimental] Prefill-Decode (P/D) Disaggregation on Intel XPU 章节，包含测试模型和启动命令。

关键文件：

- python/sglang/srt/disaggregation/nixl/conn.py（模块 传输层；类别 source；类型 core-logic；符号 _send_kvcache_generic, send_kvcache_slice）：核心修复文件，所有 dtype 变更集中在 _send_kvcache_generic 和 send_kvcache_slice 函数中。
- test/registered/disaggregation/test_disaggregation_xpu.py（模块 XPU 测试；类别 test；类型 test-coverage；符号 TestDisaggregationNixlBasic, setUpClass, test_completion_returns_text, test_completion_correct_output）：新增集成测试，验证 XPU 上 NIXL 后端可用性和基本完成正确性，与核心修复联动。

- docs/platforms/xpu.md（模块 平台文档；类别 docs；类型 documentation）：平台文档更新，添加 P/D 分解实验性支持部分，提供启动步骤和验证命令。

关键符号：_send_kvcache_generic, send_kvcache_slice, make_req_array

关键源码片段

python/sglang/srt/disaggregation/nixl/conn.py

核心修复文件，所有 dtype 变更集中在 _send_kvcache_generic 和 send_kvcache_slice 函数中。

```
def _send_kvcache_generic(
    self,
    peer_name: str,
    src_data_ptrs: list[int],
    dst_data_ptrs: list[int],
    item_lens: list[int],
    prefill_data_indices: npt.NDArray[np.int32],
    dst_data_indices: npt.NDArray[np.int32],
    dst_gpu_id: int,
    notif: str,
):
    """Generic KV cache transfer supporting both MHA and MLA architectures."""
    # 将指针列表转为 np.uint64 数组，防止 XPU 高位地址溢出 int64
    src_data_ptrs = np.array(src_data_ptrs, dtype=np.uint64)
    dst_data_ptrs = np.array(dst_data_ptrs, dtype=np.uint64)
    item_lens = np.array(item_lens, dtype=np.uint64)

    # group by indices
    prefill_kv_blocks, dst_kv_blocks = group_concurrent_contiguous(
        prefill_data_indices, dst_data_indices
    )

    logger.debug(f"sending kvcache to {peer_name} with notif {notif}")
    # ... 后续逻辑使用 uint64 类型的数组进行运算

    # 预计算 block 起始和长度时也使用 uint64
    prefill_starts = np.fromiter(
        (block[0] for block in prefill_kv_blocks), dtype=np.uint64
    )
    dst_starts = np.fromiter((block[0] for block in dst_kv_blocks), dtype=np.uint64)
    block_lens = np.fromiter(
        (len(block) for block in prefill_kv_blocks), dtype=np.uint64
    )

    for src_ptr, dst_ptr, item_len in layers_params:
        lengths = item_len * block_lens
        src_addrs.append(src_ptr + prefill_starts * item_len)
        src_lens.append(lengths)
```

```
            dst_addrs.append(dst_ptr + dst_starts * item_len)
            dst_lens.append(lengths)

        def make_req_array(addr_chunks, len_chunks, gpu):
            if not addr_chunks:
                return np.empty((0, 3), dtype=np.uint64)
            flat_addrs = np.concatenate(addr_chunks).astype(np.uint64, copy=False)
            flat_lens = np.concatenate(len_chunks).astype(np.uint64, copy=False)
            return np.column_stack(
                (
                    flat_addrs,
                    flat_lens,
                    np.full_like(flat_addrs, gpu, dtype=np.uint64),
                )
            )

        src_reqs = make_req_array(src_addrs, src_lens, self.kv_args.gpu_id)
        dst_reqs = make_req_array(dst_addrs, dst_lens, dst_gpu_id)
        # ... 后续传输逻辑
```

评论区精华

mingfeima 在 review 中建议在文档标题增加 [Experimental] 标签，以明确该功能处于实验阶段，该建议已被采纳。此外，ShangmingCai 和 mingfeima 均批准了 PR。

- 文档标题添加 [Experimental] 标签 (documentation): 已采纳，最终文档中标题变为 ## [Experimental] Prefill-Decode (P/D) Disaggregation on Intel XPU。

风险与影响

- 风险：该改动仅限于 numpy dtype 变更，不影响逻辑，因此对非 XPU 平台无副作用。但 uint64 类型可能在后续计算中与 int64 混合导致 dtype promotion 问题，需要确保所有下游操作兼容 uint64。当前测试仅覆盖 XPU，其他平台可能跳过，可能掩盖回归。
- 影响：直接影响：修复 Intel XPU 上 NIXL KV 传输可能因地址溢出导致的静默数据损坏。影响范围：仅当使用 --device xpu 且启用 --disaggregation-transfer-backend nixl 时触发。文档和测试同步更新，降低了新用户的使用门槛。
- 风险标记：平台特定修复，uint64 下游兼容性，测试仅覆盖 XPU

关联脉络

- PR #24296 [Fix] Handle nixlRemoteDisconnectError in NixlKVSender: 同属 NIXL 传输层模块，该 PR 处理连接异常，本 PR 修复指针类型，共同保障 XPU 下 NIXL 稳定性。