PR #28714 完整报告

sgl-project/sglang

[minimax-m3] Split 3/4: disagg K-only index-K transfer

合并时间：2026-06-28 13:54

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/28714

执行摘要

- 一句话：添加 MiniMax-M3 K-only 稀疏 KV 的 PD-disaggregation 传递
- 推荐动作：该 PR 设计紧凑，force_flat 参数优雅地复用 MLA 布局，值得学习。建议在合并 Split 2 后尽快 rebase 并 CI 验证。对于关注 disaggregation 或稀疏 KV 的开发人员，值得精读。

功能与动机

为了支持 MiniMax-M3 模型的 PD-disaggregation，需要将稀疏 KV 的 index-K 缓冲器通过状态通道传输。Split 3/4 聚焦于 K-only 变体（不含 value），变体（含 value）尚未支持并会主动抛出 NotImplementedError。此 PR 是增量之一，确保每个 PR 独立可审查。

实现拆解

1. 新增状态类型枚举：在 python/sglang/srt/disaggregation/base/conn.py 的 StateType 枚举中添加 MINIMAX_INDEX_K 成员，用于标识 K-only 稀疏索引缓冲区。
2. 扩展 setup_state_kv_args：在 python/sglang/srt/disaggregation/utils.py 的 setup_state_kv_args 函数中，通过 isinstance(token_to_kv_pool, MiniMaxSparseKVPool) 识别 MiniMax 稀疏池，若 index_kv_pool 存在（含 value）则抛出 NotImplementedError，若 index_k_pool 存在则调用 get_index_k_state_buf_infos 并追加一个 MINIMAX_INDEX_K 状态组件。
3. 添加 K-only 传输分支：在 python/sglang/srt/disaggregation/nixl/conn.py 和 python/sglang/srt/disaggregation/mooncake/conn.py 的 maybe_send_extra 方法中，为 StateType.MINIMAX_INDEX_K 新增 elif 分支。该分支检查 PP 和 TP 限制（PP>1 或异构 TP 不支持），然后调用 _send_kvcache_generic 并传入 force_flat=True，强制使用 MLA 风格的平面布局（单缓冲每层），避免将 K-only 列表错误拆分为 K/V 两部分。
4. 状态索引载荷处理：在 python/sglang/srt/disaggregation/decode.py 和 python/sglang/srt/disaggregation/prefill.py 中，在状态索引构建函数中添加 StateType.MINIMAX_INDEX_K 分支，复用现有的 _dsa_payload() 逻辑，因为索引行与主 KV 位于同一位置，共享页 ID。
5. 单元测试覆盖：新增 test/registered/unit/disaggregation/test_minimax_sparse_disagg_state_kv_args.py，包含两个测试：test_setup_state_kv_args_single_minimax_component 验证 K-only 池生成单个 MINIMAX_INDEX_K 组件；test_index_kv_pool_raises 验证含

value 的池会正确抛出 NotImplementedError。

关键文件：

- test/registered/unit/disaggregation/test_minimax_sparse_disagg_state_kv_args.py（模块 测试；类别 test；类型 test-coverage；符号 _make_k_only_pool, _make_kv_pool, TestMiniMaxSparseDisaggStateKvArgs, test_setup_state_kv_args_single_minimax_component）：新增单元测试覆盖 K-only 和 KV 两种稀疏池的状态配置，验证核心逻辑正确性。
- python/sglang/srt/disaggregation/mooncake/conn.py（模块 传输层；类别 source；类型 core-logic）：在 maybe_send_extra 中添加 MINIMAX_INDEX_K 分支，使用 force_flat 参数调用 _send_kvcache_generic，实现 K-only 稀疏索引的 Mooncake 传输。
- python/sglang/srt/disaggregation/nixl/conn.py（模块 传输层；类别 source；类型 core-logic）：在 maybe_send_extra 中添加 MINIMAX_INDEX_K 分支，使用 force_flat 参数调用 _send_kvcache_generic，实现 K-only 稀疏索引的 NIXL 传输。
- python/sglang/srt/disaggregation/utils.py（模块 状态路由；类别 source；类型 dependency-wiring）：在 setup_state_kv_args 中添加 MiniMaxSparseKVPool 分支，导入新池类型并根据池配置追加 MINIMAX_INDEX_K 状态组件或抛出 NotImplementedError。
- python/sglang/srt/disaggregation/decode.py（模块 状态路由；类别 source；类型 core-logic）：在状态索引构建中添加 MINIMAX_INDEX_K 分支，复用 DSA 载荷逻辑，因为索引行与主 KV 位于同一位置。
- python/sglang/srt/disaggregation/prefill.py（模块 状态路由；类别 source；类型 core-logic）：在状态索引构建中添加 MINIMAX_INDEX_K 分支，复用 DSA 载荷逻辑，与 decode 保持一致。
- python/sglang/srt/disaggregation/base/conn.py（模块 基础类型；类别 source；类型 core-logic）：新增 StateType.MINIMAX_INDEX_K 枚举成员，作为 K-only 稀疏索引状态的标识。

关键符号：setup_state_kv_args, maybe_send_extra, _send_kvcache_generic, _make_k_only_pool, _make_kv_pool, test_setup_state_kv_args_single_minimax_component, test_index_kv_pool_raises

关键源码片段

test/registered/unit/disaggregation/test_minimax_sparse_disagg_state_kv_args.py

新增单元测试覆盖 K-only 和 KV 两种稀疏池的状态配置，验证核心逻辑正确性。

```
# _make_k_only_pool: 创建仅包含 index_k 的稀疏池，用于测试
def _make_k_only_pool(start_layer: int = 0) -> MiniMaxSparseKVPool:
    # 使用 MiniMax-M3 配置：密集层 0-2，稀疏层 3-6，所有稀疏层仅 K
    dense_layer_ids = [start_layer, start_layer + 1, start_layer + 2]
    sparse_layer_ids = [start_layer + 3 + i for i in range(4)]
    end_layer = sparse_layer_ids[-1] + 1
    return MiniMaxSparseKVPool(
        size=8, page_size=4, dtype=torch.float32,
        head_num=2, head_dim=8, idx_head_dim=16,
```

```
        dense_layer_ids=dense_layer_ids, sparse_layer_ids=sparse_layer_ids,
        disable_value_sparse_layer_ids=sparse_layer_ids, # 关闭 value 层
        device="cpu", start_layer=start_layer, end_layer=end_layer,
    )

# _make_kv_pool: 创建带有 index_kv（含 value）的稀疏池，期望触发 NotImplementedError
def _make_kv_pool(start_layer: int = 0) -> MiniMaxSparseKVPool:
    dense_layer_ids = [start_layer, start_layer + 1]
    sparse_layer_ids = [start_layer + 2, start_layer + 3]
    end_layer = sparse_layer_ids[-1] + 1
    return MiniMaxSparseKVPool(
        size=8, page_size=4, dtype=torch.float32,
        head_num=2, head_dim=8, idx_head_dim=16,
        dense_layer_ids=dense_layer_ids, sparse_layer_ids=sparse_layer_ids,
        disable_value_sparse_layer_ids=[], # 保留 value，index_kv_pool 不为 None
        device="cpu", start_layer=start_layer, end_layer=end_layer,
    )

class TestMiniMaxSparseDisaggStateKvArgs(unittest.TestCase):
    def test_setup_state_kv_args_single_minimax_component(self):
        pool = _make_k_only_pool()
        kv_args = KVArgs()
        setup_state_kv_args(kv_args, pool)
        # 预期只有一组件，类型为 MINIMAX_INDEX_K
        self.assertEqual(kv_args.state_types, [StateType.MINIMAX_INDEX_K])
        self.assertEqual(len(kv_args.state_data_ptrs), 1)
        self.assertEqual(len(kv_args.state_data_ptrs[0]), pool.index_k_pool.layer_num)
        self.assertEqual(len(kv_args.state_item_lens[0]), pool.index_k_pool.layer_num)

    def test_index_kv_pool_raises(self):
        pool = _make_kv_pool()
        self.assertIsNotNone(pool.index_kv_pool)
        kv_args = KVArgs()
        # 含 value 的稀疏池应引发 NotImplementedError
        with self.assertRaises(NotImplementedError):
            setup_state_kv_args(kv_args, pool)
```

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

在 maybe_send_extra 中添加 MINIMAX_INDEX_K 分支，使用 force_flat 参数调用 _send_kvcache_generic，实现 K-only 稀疏索引的 Mooncake 传输。

```
# 在 maybe_send_extra 中处理 MINIMAX_INDEX_K 状态类型
elif st == StateType.MINIMAX_INDEX_K:
    # 暂不支持 PP>1 和异构 TP（稀疏层列表压缩导致）
    if self.pp_size is not None and self.pp_size > 1:
        raise RuntimeError(
            "PD disagg: PP>1 not supported for MiniMax sparse index yet."
        )
    if (
```

```
        target_rank_registration_info is not None
        and self.attn_tp_size != target_rank_registration_info.dst_attn_tp_size
    ):
        raise RuntimeError(
            "PD disagg: heterogeneous TP not supported for MiniMax "
            "sparse index yet."
        )
    src_indices = list(indices)
    dst_indices_local = list(dst_indices)
    # 长度不一致时截断以匹配较小方
    if len(src_indices) > len(dst_indices_local):
        src_indices = src_indices[: len(dst_indices_local)]
    elif len(src_indices) < len(dst_indices_local):
        dst_indices_local = dst_indices_local[: len(src_indices)]
    rc = (
        self._send_kvcache_generic(
            mooncake_session_id=req.mooncake_session_id,
            src_data_ptrs=src_data_ptrs,
            dst_data_ptrs=dst_data_ptrs,
            item_lens=src_item_lens,
            prefill_data_indices=np.array(src_indices, dtype=np.int32),
            dst_data_indices=np.array(dst_indices_local, dtype=np.int32),
            executor=executor,
            force_flat=True, # 强制 MLA 风格平面布局，避免 KV 拆分
        ) or rc
    )
```

python/sglang/srt/disaggregation/utils.py

在 setup_state_kv_args 中添加 MiniMaxSparseKVPool 分支，导入新池类型并根据池配置追加 MINIMAX_INDEX_K 状态组件或抛出 NotImplementedError。

```
# 在 setup_state_kv_args 中的新增分支
from sglang.srt.mem_cache.memory_pool import (
    DSATokenToKVPool,
    HybridLinearKVPool,
    MiniMaxSparseKVPool,
)

# ... 变量初始化 ...

if isinstance(token_to_kv_pool, MiniMaxSparseKVPool):
    # 如果稀疏池包含 index_kv（即含有 value 的索引），暂不支持传输
    if token_to_kv_pool.index_kv_pool is not None:
        raise NotImplementedError(
            "PD disaggregation for MiniMax sparse layers with index value "
            "(index_kv_pool) is not yet supported; only K-only sparse layers are."
        )
    # 如果只有 index_k（K-only），则获取缓冲信息并追加为 MINIMAX_INDEX_K 状态
    if token_to_kv_pool.index_k_pool is not None:
```

```
            dp, dl, il = token_to_kv_pool.get_index_k_state_buf_infos()
            append_state_component(kv_args, StateType.MINIMAX_INDEX_K, dp, dl, il)
    elif hasattr(token_to_kv_pool, "get_state_buf_infos"):
        # 原有的其他池处理逻辑（如 SWA、DSA、HybridLinear 等）
        ...
```

评论区精华

仅有一条来自 ShangmingCai 的批准评论：‘Looks good. Very clean.’，未发现争议或未解问题。

- 代码审查批准 (other): PR 获得批准，无需修改。

风险与影响

- 风险：
  - 依赖风险：PR 依赖 #28713（Split 2）中的 MiniMaxSparseKVPool，若 Split 2 未合并则导入失败。但 CI 预期会失败，计划先合并 Split 2 再 rebase。
  - 兼容性风险：新增的 MINIMAX_INDEX_K 状态类型只在 MiniMax-M3 模型路径中激活，不影响现有模型。
  - 错误处理完善：对 PP>1 和异构 TP 使用了显式 RuntimeError，避免静默数据损坏。
  - 性能影响：仅增加少量分支判断，传输路径复用现有 _send_kvcache_generic 框架。
- 影响：
  - 用户：仅 MiniMax-M3 用户可受益，且当前仅支持 K-only 变体，价值用户需等待 Split 4 的完整模型集成。
  - 系统：disaggregation 框架新增一个状态类型，nixl 和 mooncake 传输层各增加一个条件分支，后续扩展 KV 支持需修改此处。
  - 团队：代码清晰，易于审查和未来扩展。
  - 风险标记：依赖上游拆分，新状态类型，PP>1 未支持，作用域有限

关联脉络

- PR #28713 [minimax-m3] Split 2/4: mem-cache / HiCache / sparse KV pool: 此 PR 依赖 Split 2 提供 MiniMaxSparseKVPool。
- PR #27944 MiniMax-M3 full feature (original large PR): 此 PR 是原 MiniMax-M3 大 PR 的拆分部分。