PR #32837 完整报告

sgl-project/sglang

feat: support Kimi Linear PD disaggregation with DCP

合并时间：2026-07-31 17:14

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/32837

执行摘要

- 一句话：新增 PD 解耦下稠密 MLA 转 DCP 布局的 KV 传输，支持 Kimi Linear
- 推荐动作：值得精读。重点看三处设计：build_dcp_token_transfer_plan 的取模规划（虚拟页 × rank 所有权）、resolve_dcp_dst_entry_indices 的 PP 组合思路、以及传输元数据在 ZMQ 消息上的字段布局约定。后者是这次 review 的主要争论点，也是后续所有 disaggregation 元数据扩展的公共约束。

功能与动机

Kimi Linear 这类线性注意力模型在 decode 阶段需要 DCP（Data Center Parallel）布局，每个 rank 只拥有部分 token 行；而 PD 解耦要求 prefill 把 MLA KV cache 搬到 decode 侧。PR body 明确列出支持拓扑矩阵：TP/DCP1 → DCP=N 是本次新增路径，而 DCP=N → TP/DCP1、DCP=N → DCP=M（N≠M）明确不支持，说明这是为 Kimi Linear 场景定制的能力补齐，而非完整的笛卡尔积支持。PR body 还强调"DCP decode requires an MLA or hybrid-MLA KV layout; the implementation is not hardcoded to Kimi, but Kimi Linear is the acceptance target."

实现拆解

1. 传输规划层：python/sglang/srt/disaggregation/common/utils.py 新增 DCPTokenTransferPlan 冻结数据类与 build_dcp_token_transfer_plan()。它基于虚拟页（physical_page_size × dcp_size）计算 decode 侧各 rank 拥有的 token 偏移，把 prefill 物理页 token 行映射到 decode 物理页内的目标行，并校验 decode_prefix_len 与虚拟页对齐、num_kv_tokens 不超过源容量、目标页充足。规划结果 src_token_indices/dst_token_indices 一一对应，供后端直接组装传输。
2. 注册与协商层：Mooncake 与 NIXL 各自的 KVArgsRegisterInfo 新增 dst_dcp_size、dst_dcp_rank、requires_dcp_relayout、dcp_token_item_lens 等字段；common/conn.py 的 CommonKVSender 新增 requires_dcp_relayout()（同规模 DCP 走原路径，DCP1→DCP=N 且 MLA 后端走新路径，其余抛错）与 prepare_dcp_token_item_lens()（校验两侧每 token 行几何一致）；try_ensure_parallel_info() 补充 decode DCP 的运行时约束：必须 MLA/hybrid-MLA、prefill attention CP=1、两侧 page size 与 KV dtype 一致。disaggregation/prefill.py 相应把 DCP 信息写入注册消息。

3. 传输执行层：Mooncake conn.py 新增 send_kvcache_dcp()，用 group_concurrent_contiguous() 把 relayout plan 中同时连续的 src/dst 行分组，按层（dcp_token_item_lens）组装连续传输块并行下发；NIXL conn.py 同样新增 send_kvcache_dcp()，使用扁平 token-row 描述符、绕过页级 prepared handle，并新增 is_dummy_rank 标志支持某个 rank 拥有零行时只发 standalone notification 的场景。
4. PP 组合：disaggregation/utils.py 新增 resolve_dcp_dst_entry_indices()，按全局 model-layer ID 解析 decode 侧目标条目，使 PP 层划分与 DCP 上下文所有权组合；支持 prefill PP=N → decode PP=N 与 prefill PP=N → decode PP=1 两种拓扑。
5. 参数约束：arg_groups/pd_disaggregation_hook.py 增加 PD + DCP 组合校验：decode DCP 强制 chunk cache（disable_radix_cache=True）、拒绝 radix/hierarchical cache、仅允许 mooncake 或 nixl 后端、prefill 侧配 DCP 时给出性能告警（prefill 用 DCP 无收益）。
6. 测试与配套：新增 8-GPU B200 nightly 验收测试 test_kimi_linear_pd_dcp4.py，覆盖 TP4/EP4/DCP1 → TP4/DCP4、monolithic 与 PD 的 token/logprob parity、物理页 / 虚拟页 / 块边界 prompt（63/64/65/255/256/257/8191/8192/8193）、32K needle 检索、CUDA-graph 与 eager decode 混合批次；test_server_args.py 新增 5 个参数约束单元测试；test_nixl_backend_basic.py 补充 NIXL 传输测试。

关键文件：

- python/sglang/srt/disaggregation/common/utils.py（模块 解耦传输；类别 source；类型 core-logic；符号 DCPTokenTransferPlan, build_dcp_token_transfer_plan）：新增 DCPTokenTransferPlan 与 build_dcp_token_transfer_plan，是稠密 → DCP relayout 的核心规划器，定义 token 行映射与全部边界校验。
- python/sglang/srt/disaggregation/mooncake/conn.py（模块 解耦传输；类别 source；类型 core-logic；符号 send_kvcache_dcp, set_transfer_blocks, process_layer）：PD + DCP 在 Mooncake 后端的主实现，新增 send_kvcache_dcp、KVArgsRegisterInfo 的 DCP 字段与 from_zmq 协议解析，是 review 讨论字段顺序的主要位置。
- python/sglang/srt/disaggregation/nixl/conn.py（模块 解耦传输；类别 source；类型 core-logic；符号 send_kvcache_dcp）：PD + DCP 在 NIXL 后端的主实现，新增 send_kvcache_dcp 与 is_dummy_rank 零行通知逻辑，与 Mooncake 路径形成对照。
- python/sglang/srt/disaggregation/common/conn.py（模块 解耦传输；类别 source；类型 core-logic；符号 requires_dcp_relayout, prepare_dcp_token_item_lens）：新增 requires_dcp_relayout 与 prepare_dcp_token_item_lens，并补充 decode DCP 的 MLA/CP=1 等运行时约束；requires_dcp_relayout 的错误处理是 review 核心争议点。
- python/sglang/srt/disaggregation/utils.py（模块 解耦传输；类别 source；类型 core-logic；符号 resolve_dcp_dst_entry_indices）：新增 resolve_dcp_dst_entry_indices，按全局 model-layer ID 解析目标条目，实现 PP 层划分与 DCP 上下文所有权的组合。
- python/sglang/srt/arg_groups/pd_disaggregation_hook.py（模块 参数校验；类别 source；类型 core-logic）：PD + DCP 组合的参数约束入口：强制 chunk cache、仅允许 mooncake/nixl 后端、prefill DCP 性能告警。
- python/sglang/srt/disaggregation/prefill.py（模块 解耦传输；类别 source；类型 core-logic）：prefill 侧注册消息适配，携带 DCP 信息并检测对端是否需要 relayout。

- test/registered/disaggregation/test_kimi_linear_pd_dcp4.py（模块 验收测试；类别 test；类型 test-coverage；符号 TestKimiLinearPDDCP4, _monolithic_reference_args, _build_boundary_prompt, _has_eight_blackwell_gpus）：8-GPU B200 nightly 验收测试，覆盖 TP4/EP4/DCP1 → TP4/DCP4、monolithic parity、页 / 块边界、NIAH 检索与 CUDA-graph/eager 混合批次。
- test/registered/unit/server_args/test_server_args.py（模块 参数校验；类别 test；类型 test-coverage；符号 test_pd_prefill_dcp_warns_about_performance, test_pd_decode_dcp_forces_chunk_cache, test_pd_decode_dcp_rejects_unsupported_transfer_backend, test_pd_decode_dcp_rejects_radix_cache）：新增 5 个 PD + DCP 参数约束单元测试，覆盖后端限制、chunk cache 强制与 radix/hierarchical cache 拒绝。
- test/registered/unit/disaggregation/test_nixl_backend_basic.py（模块 解耦传输；类别 test；类型 test-coverage）：补充 NIXL 后端基础传输测试。
- python/sglang/srt/disaggregation/base/conn.py（模块 解耦传输；类别 source；类型 data-contract）：基础 KVArgs 增加 num_kv_tokens 透传字段。
- python/sglang/srt/disaggregation/fake/conn.py（模块 解耦传输；类别 source；类型 data-contract）：fake 后端同步适配 num_kv_tokens 字段。

关键符号：build_dcp_token_transfer_plan, send_kvcache_dcp, requires_dcp_relayout, prepare_dcp_token_item_lens, resolve_dcp_dst_entry_indices, set_transfer_blocks, process_layer

关键源码片段

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

PD + DCP 在 Mooncake 后端的主实现，新增 send_kvcache_dcp、KVArgsRegisterInfo 的 DCP 字段与 from_zmq 协议解析，是 review 讨论字段顺序的主要位置。

```
@dataclasses.dataclass
class KVArgsRegisterInfo:
    # 既有字段省略：room / endpoint / dst_port / mooncake_session_id / dst_kv_ptrs / dst_aux_
    ptrs / dst_state_data_ptrs / dst_tp_rank / dst_attn_tp_size / dst_kv_item_len / dst_state_item_
    lens / dst_state_dim_per_tensor / dst_kv_layer_ids / dst_state_layer_ids

    # PD + DCP 新增字段：decode 侧 DCP 规模与 rank、是否需要稠密 -> DCP relayout、
    # 以及每层每 token 行的字节长度（用于按层组装传输块）
    dst_dcp_size: int = 1
    dst_dcp_rank: int = 0
    requires_dcp_relayout: bool = False
    dcp_token_item_lens: Optional[List[int]] = None
    # Note: always put the staging field at the final (since the staging field is
    # optional and contains multiple inputs)
    staging: Optional[StagingRegisterInfo] = None

    @classmethod
    def from_zmq(cls, msg: List[bytes]):
        return cls(
```

```
            # 既有字段省略：msg[0:14] 的解析保持不变
            dst_state_layer_ids=(
                unpack_int_lists(msg[13], "I")
                if len(msg) > 13 and msg[13] != b""
                else []
            ),
            # 注意：msg[14:16] 属于 staging 字段，DCP 字段只能追加在 staging
            # 之后（msg[16:18]）。这保证了 staging 的 from_zmq_fields(msg, 14)
            # 固定从 14 号槽位开始解析，但代价是新字段无法直接追加在末尾，
            # 跨版本兼容性依赖"staging 永远最后"的约定（review 讨论点）
            dst_dcp_size=(
                int(msg[16].decode("ascii")) if len(msg) > 16 and msg[16] != b"" else 1
            ),
            dst_dcp_rank=(
                int(msg[17].decode("ascii")) if len(msg) > 17 and msg[17] != b"" else 0
            ),
            # Note: always put the staging field at the final
            staging=StagingRegisterInfo.from_zmq_fields(msg, 14),
        )
```

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

新增 requires_dcp_relayout 与 prepare_dcp_token_item_lens，并补充 decode DCP 的 MLA/CP=1 等运行时约束；requires_dcp_relayout 的错误处理是 review 核心争议点。

```
    def requires_dcp_relayout(self, dst_dcp_size: int, dst_dcp_rank: int) -> bool:
        # DCP 规模相同：要求两侧 rank 对齐，否则直接报错；
        # 规模一致时无需 relayout，按原有稠密路径传输即可
        if self.dcp_size == dst_dcp_size:
            if self.dcp_rank != dst_dcp_rank:
                raise RuntimeError(
                    "PD peers must connect matching DCP ranks, got "
                    f"prefill={self.dcp_rank}, decode={dst_dcp_rank}"
                )
            return False

        # prefill 为稠密（DCP1）、decode 为 DCP，且 KV 是 MLA 或 hybrid-MLA 时
        # 走新的稠密 -> DCP relayout 路径（本次 PR 的核心新增能力）
        if (
            self.dcp_size == 1
            and dst_dcp_size > 1
            and (self.is_mla_backend or self.is_hybrid_mla_backend)
        ):
            return True

        # 其余拓扑（DCP -> 稠密、DCP 规模不等）目前不支持。
        # 注意：review 指出这里直接抛 RuntimeError 会让传输 worker 崩溃，
        # 而同一个 prefill 实例可能同时服务普通 TP decode，宜改为请求级失败处理
        raise RuntimeError(
            f"Unsupported PD DCP topology: {self.dcp_size} -> {dst_dcp_size}"
```

)

评论区精华

评审由 ShangmingCai 发起并最终 APPROVED（YAMY1234 亦 LGTM），核心交锋集中在两点：

1. from_zmq 字段顺序：ShangmingCai 指出 staging 字段应物理上放在消息最后，DCP 新字段却排在 staging 之后（msg[16:18]），追加字段会破坏版本兼容，且 StagingRegisterInfo 字段数从名字无法得知、可读性差，建议后续重构元数据解释逻辑。
2. requires_dcp_relayout 的错误处理：ShangmingCai 认为直接 raise RuntimeError 会让传输 worker/ 元数据监听线程崩溃——同一个 prefill 实例可能同时配对 DCP *decode* 与普通 TP *decode*，崩溃会殃及后者；应把该请求标记失败并告警，或用 try 块包裹避免引擎崩溃。 作者 kpham-sgl 明确回复 “Will address comments in a following PR”，GPU CI 通过后合并。

- from_zmq 中 DCP 字段顺序与 staging 位置约定 (design): kpham-sgl 表示将在后续 PR 中处理；本 PR 合并时保持 msg[16:18] 的布局。
- requires_dcp_relayout 抛 RuntimeError 导致传输 worker 崩溃 (design): kpham-sgl 表示将在后续 PR 中处理；本 PR 保留抛错行为。
- 新字段追加与版本兼容（StagingRegisterInfo 展开） (design): 待后续重构；YAMY1234 未公开回应，作者确认后续 PR 处理。

风险与影响

- 风险：
  1. 传输 worker 崩溃风险：common/conn.py 的 requires_dcp_relayout() 在拓扑不匹配时抛 RuntimeError，按 review 讨论会拖垮同实例上其他正常请求（生产上 prefill 可能同时服务 DCP 与普通 *decode*）。
  2. 协议字段顺序兼容隐患：from_zmq 把 DCP 字段追加在 staging 之后，而 staging 占 msg[14:16] 两个槽位；后续追加字段或跨版本互通时极易错位。
  3. 拓扑误配缺少降级：DCP=N → TP/DCP1、DCP=N → DCP=M（N≠M）直接抛错，没有 fallback 或显式引导。
  4. 验收覆盖单一：8-GPU 端到端测试只覆盖 TP4/DCP1 → TP4/DCP4 一种组合；PR body 声明 PP 组合与更宽拓扑规则已实现，但无自动化覆盖。
  5. NIXL 零拥有行 standalone notification 是新路径，is_dummy_rank 与既有 is_dummy() 语义并存，需关注空行 rank 的完成通知时序。 - 影响：影响范围集中在 disaggregation 模块：common/mooncake/nixl 三处 conn.py、prefill.py、server_args 约束与两个测试文件。对既有非 DCP PD 行为（TP/DCP1 → TP/DCP1）保持兼容；新增能力使 Kimi Linear（以及任何 MLA/hybrid-MLA + DCP *decode* 组合）可以在 PD 解耦下工作，是切入线性注意力 + DCP *decode* 生产路径的关键一环。团队侧新增一个 est_time=1200s 的 nightly-8-gpu-b200 验收测试与 5 个参数约束单测，CI 负担可控。 - 风险标记：传输 worker 崩溃风险，协议字段顺序兼容隐患，拓扑误配缺少降级路径，验收测试覆盖单一拓扑，NIXL 零行通知为新路径

关联脉络

- PR #36025 [AMD][MORI] Deduplicate CP-replicated state transfers: 同改 disaggregation/common/conn.py、mooncake/conn.py、nixl/conn.py 的传输元数据与 wire 协议，属于同一模块的持续演进。
- PR #35840 Add PD test for inkling with mxfp8 KV: 同为 PD 解耦场景的 KV 传输测试与 retraction 状态备份修复，验证边界运行时的数据一致性。
- PR #35957 Fix recurrent state loss on decode retraction: 同属 disaggregation 数据一致性修复脉络，与 PD 传输路径共享 schedule_batch/mem_cache 相关逻辑。