PR #22665 完整报告

sgl-project/sglang

[PD] MORI-IO: Add state transfer, inline transfer model, and high-concurrency fixes

合并时间：2026-05-09 07:07

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/22665

执行摘要

- 一句话：MORI-IO 状态转移与高并发修复
- 推荐动作：值得精读。本 PR 展示了 RDMA 传输后端的架构演进：从工作线程模型到内联发布的权衡、状态转移的索引设计、以及高并发下的锁优化策略。对于从事分布式推理或高性能网络传输的工程师极具参考价值。

功能与动机

MORI-IO 作为基于 RDMA 的 KV 传输后端在 #14626 中引入，但缺少状态转移且在高并发下存在性能瓶颈与正确性问题。本 PR 填补该空白，并重构传输架构以提升吞吐和稳定性。

实现拆解

1. 状态转移支持：在 MoriKVManager 中新增 send_state() 方法，根据 state_type 分发到 _send_mamba_state() 或 _send_swa_nsa_state()。TransferInfo 增加 dst_state_indices 字段，KVArgsRegisterInfo 增加 dst_state_item_lens 和 dst_state_dim_per_tensor。
2. 内联传输模型重构：移除原来的预填充工作线程队列，将传输提交内联到 add_transfer_request()，RDMA 发布在锁外进行；MoriKVSender.poll() 通过 _all_transfers_finished() 轮询完成状态。引入 GroupedIndexPlan 和 BatchTransferPlan 数据类预计算偏移，减少逐层开销。
3. AUX 数据传输优化：默认仍用 ZMQ TCP (send_aux_tcp())，新增可选 RDMA 路径 (send_aux_rdma())，通过环境变量 SGLANG_MORI_SEND_AUX_RDMA=1 启用，并自动回退到 TCP。添加 _connect_threadsafe() 支持线程安全连接复用。
4. 多个错误修复：修正 GQA/MQA 下 prefill_tp_size > decode_tp_size 的 TP 切片映射；在 _handle_transfer_message() 中增加陈旧元数据防护；update_status() 状态机防止 Failed 被覆盖；_compute_prefill_unique_rank() 正确编码 TP/PP/CP 秩；TCP 连接复用避免端口耗尽。
5. 测试配套：新增 test/registered/amd/disaggregation/test_mori_transfer_engine_e2e.py，继承 PDDisaggregationServerBase，启动双节点 PD 服务器并通过 HTTP 请求验证端到端传输正确性。

关键文件：

- python/sglang/srt/disaggregation/mori/conn.py（模块 传输后端；类别 source；类型 core-logic；符号 _normalize_state_indices, GroupedIndexPlan, from_groups, materialize）：核心变更文件，实现状态转移、内联传输模型、TP 切片修复、AUX 路径、连接复用等所有逻辑。改动量 +590/-163。
- test/registered/amd/disaggregation/test_mori_transfer_engine_e2e.py（模块 传输引擎；类别 test；类型 test-coverage；符号 MoriTransferEngineBase, setUpClass, tearDownClass, _shift_ports）：新增的端到端测试，验证 MORI 传输引擎在双节点 PD 配置下的基本功能，确保状态转移和 KV 传输正确性。

关键符号：send_state, _send_mamba_state, _send_swa_nsa_state, add_transfer_request, MoriKVSender.poll, GroupedIndexPlan.from_groups, BatchTransferPlan.empty, TransferTarget, update_status, _connect_threadsafe, _compute_prefill_unique_rank, _handle_transfer_message, send_aux_tcp, send_aux_rdma

关键源码片段

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

核心变更文件，实现状态转移、内联传输模型、TP 切片修复、AUX 路径、连接复用等所有逻辑。改动量 +590/-163。

```
# 从 MORI-IO 的 TransferInfo 和 KVArgsRegisterInfo 中增加状态索引字段
@dataclasses.dataclass
class TransferInfo:
    room: int
    endpoint: str
    dst_port: int
    engine_key: str
    dst_kv_indices: npt.NDArray[np.int32]
    dst_aux_index: int
    dst_state_indices: npt.NDArray[np.int32] # 新增：目标状态索引数组
    required_dst_info_num: int
    is_dummy: bool

    @classmethod
    def from_zmq(cls, payload: List[bytes]) -> TransferInfo:
        # ... 解析前 6 个字段后，处理第 7 个字段（索引 6）作为状态索引
        if len(payload) > 6 and payload[6]:
            dst_state_indices = np.frombuffer(payload[6], dtype=np.int32)
        else:
            dst_state_indices = np.array([], dtype=np.int32)
        # 返回完整对象
        return cls(room=room, endpoint=endpoint, ..., dst_state_indices=dst_state_indices)


# GroupedIndexPlan 用于预计算批次传输的索引偏移
@dataclasses.dataclass(frozen=True)
class GroupedIndexPlan:
```

```
    src_starts: List[int]
    dst_starts: List[int]
    counts: List[int]

    @classmethod
    def from_groups(
        cls, src_groups: List[List[int]], dst_groups: List[List[int]]
    ) -> GroupedIndexPlan:
        # 将每个 group 的首个索引作为 start，长度作为 count
        return cls(
            src_starts=[int(g[0]) for g in src_groups],
            dst_starts=[int(g[0]) for g in dst_groups],
            counts=[len(g) for g in src_groups],
        )


# BatchTransferPlan 聚合多个 TransferTarget，支持 batch 提交 RDMA
@dataclasses.dataclass(frozen=True)
class BatchTransferPlan:
    all_transfer_targets: List[TransferTarget]

    @classmethod
    def empty(cls) -> BatchTransferPlan:
        return cls(all_transfer_targets=[])

    def materialize(self) -> List[TransferTarget]:
        return list(self.all_transfer_targets)
```

test/registered/amd/disaggregation/test_mori_transfer_engine_e2e.py

新增的端到端测试，验证 MORI 传输引擎在双节点 PD 配置下的基本功能，确保状态转移和 KV 传输正确性。

```
class MoriTransferEngineBase(PDDisaggregationServerBase):
    port_delta = 0
    prefill_tp = 1
    decode_tp = 1
    decode_base_gpu_id = 1
    required_gpus = 2

    @classmethod
    def setUpClass(cls):
        # 检查 GPU 可用性，跳过不满足条件
        if torch.cuda.device_count() < cls.required_gpus:
            raise unittest.SkipTest(...)
        super().setUpClass()
        # 固定使用 MORI 后端
        cls.transfer_backend = ["--disaggregation-transfer-backend", "mori"]
        # 读取环境变量获取 RDMA 设备，否则无 RDMA 运行
        rdma_env = os.environ.get("SGLANG_TEST_RDMA_DEVICE")
```

```
        if rdma_env:
            cls.rdma_devices = ["--disaggregation-ib-device", rdma_env]
        else:
            cls.rdma_devices = []
        # 启动预填充和解码服务器，等待健康，然后启动负载均衡
        cls.start_prefill()
        cls.start_decode()
        cls.wait_server_ready(cls.prefill_url + "/health", ...)
        cls.wait_server_ready(cls.decode_url + "/health", ...)
        cls.launch_lb()

    def _assert_generate_smoke(self):
        # 发送 POST 请求到负载均衡器，验证生成输出非空
        resp = requests.post(
            cls.lb_url + "/v1/chat/completions",
            json={"model": cls.model, "messages": [{"role": "user", "content": "Hello"}]},
            timeout=60,
        )
        self.assertEqual(resp.status_code, 200)
        self.assertIn("choices", resp.json())
```

评论区精华

审阅者 ShangmingCai 询问是否有 AMD CI 测试，维护者 HaiShaw 回应「Coming next」。其余审查仅包括 LGTM 批准，无深层技术争论。

- AMD CI 测试覆盖 (testing): HaiShaw 回应“Coming next”，表明后续会加入 AMD CI 测试。

风险与影响

- 风险：
  1. 状态转移正确性风险：SWA/NSA 状态转移尚不支持 TP 不匹配的非 MLA 注意力，可能与某些模型不兼容。
  2. 内联传输模型的调度压力：poll() 循环在 caller 线程中运行，若传输量大可能阻塞主循环，需监控实际部署中的延迟抖动。
  3. AUX RDMA 实验性：环境变量 SGLANG_MORI_SEND_AUX_RDMA 可能造成解码暂停（受 AINIC 影响），默认 TCP 路径更稳定。
  4. 并发竞争：锁范围的减小虽然提升并发，但若 transfer_lock 与状态机交互不当，仍可能引入 subtle race。
  5. 测试覆盖有限：仅 2 个文件，且 E2E 测试只在 2-GPU 配置下运行，未覆盖 TP 不匹配、高并发压力等场景。 - 影响：对用户：PD 分离式推理的用户可获得状态转移支持（Mamba/SWA/NSA 模型），高并发下吞吐提升约 10%（见 benchmark），且连接管理更稳定。对系统：内联模型减少线程开销，降低 OOM 风险；但需要 consumer 主动 poll，增加单线程负担。对团队：维护点集中到 conn.py，新架构更清晰，但需持续监控性能退化。 - 风险标记：状态转移 TP 不匹配限制，AUX RDMA 实验性可能解码暂停，内联 poll 引入主循环风险，高并发下锁竞争隐患，测试覆盖未包含 TP 变化场景

关联脉络

- 暂无明显关联 PR