

PR #39584 完整报告

sgl-project/sglang

[Fix][PD] Give prefill and decode their own RDMA NICs in disaggregation tests

合并时间: 2026-09-16 06:07

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/39584>

执行摘要

- 一句话: PD 测试按 GPU 索引分配 RDMA 网卡, 消除跨 socket 超时
- 推荐动作: 值得精读。虽然是纯测试代码, 但它是一个“静默性能劣化”的完整诊断案例: 从症状 (首请求 300s 超时且测试不红) 到根因 (CUDA_VISIBLE_DEVICES 未参与、盲选 fallback 跨 socket) 再到三组对照验证, 方法论可以复用到其他 CI 超时排查。值得关注的设计决策: 绝对 GPU id 映射、删掉静默回退分支、把三个 helper 合并为一个函数。建议后续跟进两件事: 确认 dsv4/hisparse 等覆盖面较小的文件在新映射下无明显回归, 并考虑把 inkling 那类“range 来自 COMMON_ARGS”的注释提升为 fixture 约定。

功能与动机

PR body 明确指出根因: PD 测试用 `--base-gpu-id` 把 decode 侧偏移到后半段 GPU, 但 `get_rdma_devices_args()` 从 `CUDA_VISIBLE_DEVICES` 推导 NIC, 而 PD 测试从不设置该环境变量, 于是每次调用都落入盲选 fallback, 两侧共享同一对 NIC (`mlx5_1`, `mlx5_3`), 两者都在 NUMA 0, decode 却跑在 NUMA 1 的 GPU 4-5 上, `nvidia-smi topo -m` 对每条路径都报告 SYS。更隐蔽的是“只有第一个请求烧满超时, 其余请求成功, 测试保持绿色、成本不可见”。作者提供了量化证据: 同一测试类仅设备列表不同, decode NIC 从 (`mlx5_1`, `mlx5_3`) (跨 socket) 改为 `mlx5_4` 后, eval 延迟从 301.33s 降到 2.10s。

实现拆解

变更入口是 `python/sglang/test/server_fixtures/disaggregation_fixture.py`, 随后向 11 个覆写 `start_prefill/start_decode` 的测试文件同步替换。分四步:

1. 重写 `get_rdma_devices_args()` (fixture 内模块级函数): 新增可选参数 `gpu_indices`, 缺省时回退到 `CUDA_VISIBLE_DEVICES`; GPU id 先经 `sorted({int(g) for g in ...})` 归一化; 核心映射改为 `min(gpu // gpus_per_rdma, n_rdma - 1)`, 直接用节点级绝对 GPU id 整除映射, 不再用组内相对索引换算。同时删除两个问题分支: `len(gpu_indices) > 4` 的静默回退 (它让 8 卡环境永远失去亲和性) 和仅发警告的 `base_rdma_group` 校验。`total_gpus` 探测失败时用 `max(8, gpu_indices[-1] + 1)` 兜底, 保证映射不为 0。
2. 新增 `rdma_devices_for()` 夹具方法: `PDDisaggregationServerBase.rdma_devices_for(cls, gpu_indices)` 在 CI 下返回 `["--disaggregation-ib-device", get_rdma_devices_args(gpu_indices)]`, 非 CI 直接返回静态 `cls.rdma_devices` (行为与原先一致)。fixture 默认的 `start_prefill/start_decode` 分别传 `range(prefill_tp_size)` 与 `range(decode_base_gpu_id, decode_base_gpu_id + decode_tp_size)`, 与

--base-gpu-id 语义对齐。

3. 批量替换 42 处覆写点：test_disaggregation_different_tp.py (8 组 TP 2/4 组合)、test_epd_disaggregation.py (encode/prefill/decode 三角色在 GPU 0/1/2 各自独立, prefill 传 range(1,2)、decode 传 range(2,3))、test_disaggregation_hybrid_attention.py、test_disaggregation_dp_attention.py (用 PREFILL_DP_SIZE/DECODE_DP_SIZE 类常量推导 range)、test_disaggregation_nixl.py (基类 NixlPDDisaggregationServerBase 也换用)、test_disaggregation_pp.py、test_disaggregation_inkling_mxfp8.py (范围由 COMMON_ARGS 中的 --tp 2 决定, prefill range(2)、decode range(2,4), 并加注释说明)、test_disaggregation_aarch64.py、test_disaggregation_decode_offload.py、test_disaggregation_dsv4.py、test_disaggregation_hispase.py。
4. 验证配套：base-c-test-8-gpu-h20 三个 partition 全绿, different_tp 1184s → 859s, StagingRadixPrefillLargerTP 的 eval 从 301.3s 到 3.87s; 另在 8-gpu-h200、4-gpu-h100、8-gpu-b300、4-gpu-b200 与 2-gpu-large 跑通, 并对 hybrid_attention、hispase、decode_radix_cache_swa、inkling_mxfp8、dwdp_gpt_oss、decode_offload、aarch64 执行 /rerun-test 全部通过。AMD 测试不受影响：其各类均自行实现 start_prefill/start_decode 并直接读 cls.rdma_devices, 不经过本 PR 改动的 fixture 默认路径。

关键文件：

- python/sglang/test/server_fixtures/disaggregation_fixture.py (模块 测试夹具；类别 test；类型 test-coverage；符号 rdma_devices_for, get_rdma_devices_args, _pick_default_pair)：核心变更文件：重写 get_rdma_devices_args() 改用绝对 GPU id 映射, 新增 rdma_devices_for() 夹具方法并在默认 start_prefill/start_decode 中接入, 删除盲选回退与 base_rdma_group 校验, 函数从 141 行压缩到 57 行。
- test/registered/disaggregation/test_disaggregation_different_tp.py (模块 TP 差异；类别 test；类型 test-coverage)：直接受益最大的测试 (1184s → 859s)，8 组 TP 组合的 start_prefill/start_decode 全部换用 rdma_devices_for(range(...))。
- test/registered/disaggregation/test_epd_disaggregation.py (模块 编码拆分；类别 test；类型 test-coverage)：EPD 三角色 (encode/prefill/decode) 各占不同 GPU, 替换为 rdma_devices_for(range(1,2))/range(2,3) 等, 验证多角色场景下的映射正确性。
- test/registered/disaggregation/test_disaggregation_hybrid_attention.py (模块 混合注意力；类别 test；类型 test-coverage)：多组 GDN/Mamba/DP 组合, decode 侧用 range(4,8)/range(2,4) 精确覆盖 --base-gpu-id 偏移；同时被 /rerun-test 验证。
- test/registered/disaggregation/test_disaggregation_dp_attention.py (模块 DP 注意力；类别 test；类型 test-coverage)：用类常量 PREFILL_DP_SIZE/DECODE_DP_SIZE 推导 GPU 范围, 是“按配置推导 range”的参考写法。
- test/registered/disaggregation/test_disaggregation_nixl.py (模块 NIXL 后端；类别 test；类型 test-coverage)：NIXL 后端基类 NixlPDDisaggregationServerBase 也接入 rdma_devices_for, 证明该接口对非 Mooncake 后端同样生效。
- test/registered/disaggregation/test_disaggregation_pp.py (模块 PP 拆分；类别 test；类型 test-coverage)：PP 拆分用例同步替换, 覆盖 tp 4/2 与 decode 偏移组合。

- test/registered/disaggregation/test_disaggregation_inkling_mxfp8.py (模块 MXFP8 用例; 类别 test; 类型 test-coverage) : 特殊点: tp 来自 COMMON_ARGS (--tp 2), 提交 efca03f 专门修正该文件的 range, 并加注释说明依赖关系。
- test/registered/disaggregation/test_disaggregation_decode_offload.py (模块 解码卸载; 类别 test; 类型 test-coverage) : HiCache 解码卸载用例, prefill 单卡 range(1)、decode range(1,2), 是最小规模映射的样例。
- test/registered/disaggregation/test_disaggregation_aarch64.py (模块 ARM 测试; 类别 test; 类型 test-coverage) : ARM 节点用例 range(2)/range(2,4), 验证跨架构 CI lane 兼容。
- test/registered/disaggregation/test_disaggregation_dsv4.py (模块 DSV4 用例; 类别 test; 类型 test-coverage) : DSV4 传输用例同步替换 (2 行), 与 PD 传输性能优化方向相关。
- test/registered/disaggregation/test_disaggregation_hispase.py (模块 稀疏注意力; 类别 test; 类型 test-coverage) : HiSparse 用例同步替换 (2 行), 被 /rerun-test 验证通过。

关键符号: rdma_devices_for, get_rdma_devices_args

关键源码片段

python/sglang/test/server_fixtures/disaggregation_fixture.py

核心变更文件: 重写 get_rdma_devices_args() 改用绝对 GPU id 映射, 新增 rdma_devices_for() 夹具方法并在默认 start_prefill/start_decode 中接入, 删除盲选回退与 base_rdma_group 校验, 函数从 141 行压缩到 57 行。

```
# python/sglang/test/server_fixtures/disaggregation_fixture.py
```

```
@classmethod
```

```
def rdma_devices_for(cls, gpu_indices) -> list:
```

```
    """返回某个服务进程实际占用 GPU 对应的 `--disaggregation-ib-device` 参数。
```

```
    非 CI 环境沿用静态配置 `cls.rdma_devices` (行为与旧代码一致);
```

```
    CI 环境按绝对 GPU id 实时推导, 让 prefill 与 decode 各用各的 NIC,
```

```
    避免 decode 侧 (`--base-gpu-id` 偏移) 重新映射回 NUMA 0 的第一批网卡。
```

```
    """
```

```
    if not is_in_ci():
```

```
        return cls.rdma_devices
```

```
    return ["--disaggregation-ib-device", get_rdma_devices_args(gpu_indices)]
```

```
def get_rdma_devices_args(gpu_indices=None) -> str:
```

```
    """按节点级绝对 GPU 索引选择 RDMA NIC, 返回逗号分隔的设备名。
```

```
    旧实现从 `CUDA_VISIBLE_DEVICES` 推导, 且 8 卡环境会落入“首对网卡”盲选,
```

```
    导致 PD 测试两侧共用同一对 NIC、跨 socket 传输烧满 300s 超时。
```

```
    新实现由调用方传入该服务实际占用的 GPU id (与 `--base-gpu-id` 对齐)。
```

```
"""
```

```
def _parse_list_env(var_name: str):
    # 解析逗号分隔的环境变量，空值统一返回 None
    val = os.getenv(var_name)
    items = [x.strip() for x in (val or "").split(",") if x.strip()]
    return items or None

# 设备来源优先级：环境变量 > 自动探测 > 硬编码兜底
rdma_all_devices = (
    _parse_list_env("SGLANG_CI_RDMA_ALL_DEVICES")
    or _get_available_ib_devices()
    or [f"mlx5_roce{i}" for i in range(8)]
)
n_rdma = len(rdma_all_devices)

# 未显式传参时回退到 CUDA_VISIBLE_DEVICES，并统一排序去重
if gpu_indices is None:
    gpu_indices = _parse_list_env("CUDA_VISIBLE_DEVICES") or []
try:
    gpu_indices = sorted({int(g) for g in gpu_indices})
except ValueError:
    warnings.warn(f"Invalid GPU indices: {gpu_indices}")
    gpu_indices = []
if not gpu_indices:
    # 没有任何可用索引时只能取首对 NIC（原盲选 pair 兜底）
    return ",".join([rdma_all_devices[0], rdma_all_devices[n_rdma // 2]])

# 用节点 GPU 总数估算每个 NIC 服务的 GPU 数；探测失败时按最大索引兜底
try:
    import torch

    total_gpus = torch.cuda.device_count()
except Exception:
    total_gpus = 0
total_gpus = total_gpus or max(8, gpu_indices[-1] + 1)

gpus_per_rdma = max(1, total_gpus // n_rdma)
# 关键点：用绝对 GPU id 整除映射，而不是组内相对索引，
# 这样 decode 侧（GPU 4-5）会落到后半段 NIC（如 mlx5_4），与 NUMA 拓扑对齐
devices = [
    rdma_all_devices[min(gpu // gpus_per_rdma, n_rdma - 1)] for gpu in gpu_indices
]
# 去重并保持顺序，同一侧不重复传同一 NIC
resolved = ",".join(dict.fromkeys(devices))
logger.warning(
    "RDMA for gpus=%s: total_gpus=%d n_rdma=%d -> %s",
    gpu_indices,
    total_gpus,
```

```
n_rdma,  
resolved,  
)  
return resolved
```

评论区精华

本 PR 没有人工 review 评论 (comments_count=3 均为 CI bot 对 /rerun-test 的响应)，核心论证全部写在 PR body 里，质量很高，值得提炼：

- 根因链条完整：CUDA_VISIBLE_DEVICES 未设置 → 盲选 fallback → 共享 (mlx5_1, mlx5_3) → 全路径 SYS 拓扑 → 首次 KV 传输 300s 超时。
- 三组对照实验锁定变量：同一测试类只改设备列表，8 设备自动检测 2.47s、4 设备修复前 301.33s、4 设备修复后 2.10s。
- 作者明说此类问题难以被发现：“只有第一个请求烧满超时，其余请求成功，测试保持绿色、成本不可见”——这是测试基建中典型的静默膨胀。
- commit efca03f 是作者自审修正：inkling_mxfp8 的 tp 来自 COMMON_ARGS 而非硬编码，GPU 范围最初推断错误，已改为 range(2)/range(2,4)。
- inkling_mxfp8 的 GPU 范围推断自审修正 (correctness)：采用 range(2) 与 range(2, 4)，并在代码中加注说明该依赖关系。
- 盲选 fallback 与跨 socket 超时的量化验证 (design)：确认 NIC 映射必须基于绝对 GPU id，并删除 len(gpu_indices) > 4 的静默回退分支。

风险与影响

- 风险：
 - 拓扑假设：新映射假定 NIC 均匀分摊 GPU (gpus_per_rdma = total_gpus // n_rdma)，对非对称 NUMA/NIC 布局的机器仍可能选错 NIC；但相比原先必错（固定首对 NIC）已有本质改善，且映射结果会打印 warning 日志便于核对。
 - 隐式耦合：test_disaggregation_inkling_mxfp8.py 的 range 依赖 COMMON_ARGS 中的 -tp 2，若 COMMON_ARGS 调整需同步改 range，属隐蔽耦合点。
 - 覆盖面：dsv4、hispase、decode_offload、aarch64 等文件改动最小（各 2 行），未逐一给出实测数据，主要依赖 base-c 全绿兜底。
 - 无生产代码变更：所有改动限于 python/sglang/test/server_fixtures 与 test/registered 下，不影响运行时行为；非 CI 环境路径完全不变 (rdma_devices_for 直接返回 cls.rdma_devices)。
- 影响：
 - 对用户 / 系统：零生产影响，全部改动在测试侧。
 - 对 CI：8-gpu-h20 lane 的 test_disaggregation_different_tp.py 单测耗时降约 27% (1184s → 859s)，消除了 PD 测试第一个请求隐藏的 300s 超时等待；同类收益预计覆盖所有走 fixture 默认路径的 Mooncake PD 测试。
 - 对团队：为后续 PD 测试建立新约定——凡覆写 start_prefill/start_decode 的类都应改用 cls.rdma_devices_for(range(...)) 并按实际 GPU 占用传参，否则会退回盲选 pair；

fixture 默认实现已示范正确用法。

- 对可观测性：删除的 `base_rdma_group` 警告与 `>4 GPU` 静默回退使配置路径更清晰，映射日志（RDMA for gpus=...）让网卡选择可审计。
- 风险标记：测试基础设施变更，NIC 映射依赖均匀拓扑假设，INKLING 用例与 `COMMON_ARGS` 隐式耦合，无生产代码路径变更

关联脉络

- PR #39611 [CI][Disaggregation] Fix EADDRINUSE flake in `test_disaggregation_dwdp_gpt_oss`: 同为 8-gpu disaggregation 测试稳定性修复；本 PR merge main 时还解决了与该文件相关的冲突。
- PR #39544 [misc] Trim redundant variants from the 8-gpu-h20 disaggregation test suite: 同 lane 耗时优化，目标一致（缩短 8-gpu-h20 套件运行时间），与本 PR 的 1184s → 859s 收益互补。
- PR #39600 [CI] Fix SWA decode radix cache NIXL import: 同 lane 的 CI 稳定性修复，且本 PR 的 `/rerun-test` 也覆盖了 `test_disaggregation_decode_radix_cache_swa.py`。
- PR #38984 [Intra-node PD][DSV4] Pack all layers into one batch for `INTRA_NODE_NVLINK` path: PD KV 传输性能优化方向；本 PR 修改的 `test_disaggregation_dsv4.py` 与该 DSV4 传输链路相关。