PR #31854 完整报告

sgl-project/sglang

perf(diffusion): CUDA-IPC zero-staging all-to-all for 2-rank Ulysses

合并时间：2026-07-31 19:35

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/31854

执行摘要

- 一句话：2-rank Ulysses 换用 CUDA-IPC 零拷贝 A2A
- 推荐动作：值得精读，尤其是四个设计决策：GPU 侧序列计数器同步原语（bump/spin 内核对）替代 NCCL rendezvous；CUDA graph 捕获安全性的系统处理（capture 内 Miss 返回 None 回退 NCCL）；超时双端退休机制避免进程组失同步；用独立微基准明确 2-rank 边界并留下完整的扩展性论证。测试设计（断言传输真正 engage、平台跳过、超时路径覆盖）也是可复用的范式。建议关注默认开启后的线上行为，以及 staging 淘汰顺序一致性这一隐式契约的后续演进。

功能与动机

PR body 指出：在 ulysses degree 2 单节点下，每个 transformer 层要付出 4 次 NCCL all-to-all（独立的 q/k/v 输入 A2A 加输出 A2A），每次都要一次 rendezvous 加多次全张量 staging 拷贝（transpose/contiguous/cat），profiling qwen-image 显示 A2A 数据面而非传输本身主导通信成本。关联 Issue #31852 补充了量化证据：每个 SP collective 都要付一次新的 NCCL rendezvous，单次 all-to-all 平均约 340μs，而 NVLink 纯传输只要约 25μs。本 PR 的目标是让每个字节恰好移动一次：从投影输出直接写入对端消费缓冲区，同时消除 NCCL 的 rendezvous 开销。

实现拆解

实现分为六个步骤：

1. 新增 CUDA-IPC 传输层（python/sglang/multimodal_gen/runtime/distributed/device_communicators/ipc_a2a.py）：IpcA2AState 单例 IPC_A2A 管理双缓冲 staging 槽位、序号旗标与超时状态。init() 校验 can_device_access_peer 并调用 cudaDeviceEnablePeerAccess，用 load_inline JIT 编译 spin_wait/bump_signal 一对内核；_share() 通过 reduce_tensor + broadcast_object_list 交换 IPC handle，并在本端设备上下文重建张量（vLLM custom-allreduce 风格）。
2. USPAttention A2A 家族接入（python/sglang/multimodal_gen/runtime/layers/usp.py + runtime/layers/attention/layer.py）：新增 _ipc_ready_group()、_ipc_varlen_fast()、_ipc_input_a2a_qkv()、_ipc_input_a2a_qkv_segmented() 四个钩子，在每个 input/output/varlen A2A 入口优先尝试 IPC 路径，返回 None 时静默回落原有 NCCL 逻辑。USPAttention.forward 增加 qkv_pre_all_to_all 参数，防止已由分段交换完成 gather 的 q/k/v 被二次 A2A。

3. AllToAll4D 接入（runtime/distributed/device_communicators/base_device_communicator.py）：新增 _ipc_all_to_all_4d()，在 AllToAll4D.forward 的 world_size == 2 分支命中 IPC 路径，覆盖 stacked-qkv 的 UlyssesAttention 用户（hunyuanvideo、wan-VSA、zimage 次级注意力）。
4. qwen joint attention 零拷贝变体（runtime/models/dits/qwen_image.py）：_ipc_input_a2a_qkv_segmented 直接按 join_seqs 布局 [txt_real | img | txt_pad] 切分读写，staging 缓冲区本身就是汇聚后的 q/k/v，省掉逐投影 joint cat 与中间 gather 拷贝；seg_qkv 命中时向 attention 传递 qkv_pre_all_to_all=True 短路第二次 A2A。
5. 可靠性与配置（envs.py、runtime/managers/gpu_worker.py）：新增 SGLANG_DIFFUSION_IPC_A2A（默认开）、TIMEOUT_MS（默认 10000）、MAX_BUFFERS（默认 16）三个环境变量；gpu_worker.execute_forward 在请求边界调用 IPC_A2A.check_timeout() 做设备侧看门狗读取，超时则双端退休并抛错。
6. 测试配套：新增 2-GPU parity 测试 test/single_test_file/test_ipc_a2a_2_gpu.py（NCCL 与 IPC 双路逐位比对，覆盖 USPAttention A2A 家族、AllToAll4D、staging 淘汰与超时双端退休），并在 test/server/gpu_cases.py 注册进 2-gpu 套件；测试会断言传输真正 engage（inited/staging 非空）而非两臂静默都回退，并对非 CUDA 平台跳过。

关键文件：

- python/sglang/multimodal_gen/runtime/distributed/device_communicators/ipc_a2a.py（模块 IPC 通信；类别 source；类型 core-logic；符号 _Unsupported, IpcA2AState, init, _share）：新增的核心传输层：CUDA-IPC 双缓冲 staging 交换、GPU 侧 bump/spin 同步内核、超时看门狗与双端退休机制，是整个 PR 的心脏。
- python/sglang/multimodal_gen/runtime/layers/usp.py（模块 序列并行；类别 source；类型 core-logic；符号 _ipc_ready_group, _ipc_varlen_fast, _ipc_input_a2a_qkv, _ipc_input_a2a_qkv_segmented）：USPAttention 全家桶的接入点：新增四个 IPC 钩子并把 input/output/varlen A2A 入口改为先试 IPC 后回退 NCCL，覆盖 14 个模型。
- python/sglang/multimodal_gen/test/single_test_file/test_ipc_a2a_2_gpu.py（模块 双卡测试；类别 test；类型 test-coverage；符号 _worker, both_paths, TestIpcA2ATwoGpu, test_ipc_matches_nccl_bitwise）：新增 2-GPU parity 测试：NCCL 与 IPC 双路逐位比对整个 A2A 家族，并覆盖 staging 淘汰、超时双端退休与平台跳过，是该传输能默认开启的底气。
- python/sglang/multimodal_gen/runtime/distributed/device_communicators/base_device_communicator.py（模块 设备通信；类别 source；类型 core-logic；符号 _ipc_all_to_all_4d）：为 AllToAll4D 增加 2-rank IPC 快速路径，覆盖 stacked-qkv 的 UlyssesAttention 用户（hunyuanvideo、wan-VSA、zimage 次级注意力）。
- python/sglang/multimodal_gen/runtime/models/dits/qwen_image.py（模块 模型适配；类别 source；类型 data-contract）：qwen joint attention 接入零拷贝分段交换：直接按 join_seqs 布局读写 staging，跳过逐投影 joint cat 与中间 gather，并传递 qkv_pre_all_to_all 短路第二次 A2A。
- python/sglang/multimodal_gen/runtime/layers/attention/layer.py（模块 注意力层；类别 source；类型 core-logic）：USPAttention.forward 新增 qkv_pre_all_to_all 参数与 _ipc_input_a2a_qkv 合并交换，并修正两个输入 A2A 站点中 fallback 站点未短路导致的二

次 A2A 崩溃。

- python/sglang/multimodal_gen/envs.py（模块 环境变量；类别 source；类型 configuration）：新增三个环境变量控制传输的开关、超时上限与 staging 缓冲数量上限；最终默认开启。曾因 dataclass stub 与 resolver 双真值源不一致导致默认值漂移。
- python/sglang/multimodal_gen/runtime/managers/gpu_worker.py（模块 工作进程；类别 source；类型 dependency-wiring）：在请求边界调用 IPC_A2A.check_timeout() 读取设备侧超时旗标并触发双端退休，是超时降级链路的宿主侧闭环。
- python/sglang/multimodal_gen/test/server/gpu_cases.py（模块 测试编排；类别 test；类型 test-coverage）：将 2-GPU parity 测试注册进 CI 的 2-gpu 套件并配置 240s 超时，保证传输路径有持续回归保护。

关键符号：IpcA2AState.init, IpcA2AState.get_staging, IpcA2AState.exchange, IpcA2AState.next_slot, IpcA2AState.signal, IpcA2AState.wait, IpcA2AState.check_timeout, ipc_a2a_ready, _ipc_ready_group, _ipc_varlen_fast, _ipc_input_a2a_qkv, _ipc_input_a2a_qkv_segmented, _ipc_all_to_all_4d

关键源码片段

python/sglang/multimodal_gen/runtime/distributed/device_communicators/ipc_a2a.py

新增的核心传输层：CUDA-IPC 双缓冲 staging 交换、GPU 侧 bump/spin 同步内核、超时看门狗与双端退休机制，是整个 PR 的心脏。

```
  # 每次调用交替使用 0/1 两个槽位；槽位复用由中间的 spin_wait 排序，
  # 保证本端重写某个槽位前，对端已经读完该槽位。
  def exchange(self, group, send, recv_shape):
      """对称交换：把本端连续张量 send 写入对端 staging 槽位，返回本端
      staging 槽位按 recv_shape 的视图。CUDA graph 捕获期间缓存 Miss 时
      返回 None，调用方回落 NCCL。"""
      n_send = send.numel()
      n_recv = 1
      for v in recv_shape:
          n_recv *= v
      pair = self.get_staging(n_recv, n_send, send.dtype, group)
      if pair is None:
          return None
      local, peer = pair
      slot = self.next_slot()
      # 对端映射缓冲区写入本端数据：对 2-rank 来说这是唯一一次远程写
      peer[slot].narrow(0, 0, n_send).copy_(send.view(-1), non_blocking=True)
      self.signal_and_wait()
      return local[slot].narrow(0, 0, n_recv).view(recv_shape)

  def get_staging(self, n_local, n_peer, dtype, group):
      """本地 n_local 元素缓冲（对端写这里）＋对端映射的 n_peer 缓冲（本端写）。
      创建是成对集合操作，两端必须在同一调用点命中同一 key；capture 期间
      Miss 返回 None——IPC handle 交换含分配与广播，capture 内非法。"""
```

```
    key = (n_local, n_peer, dtype)
    pair = self.staging.get(key)
    if pair is None:
        if torch.cuda.is_current_stream_capturing():
            return None # 调用方回退 NCCL，图烘焙 NCCL 路径；预热 key 保持 IPC 快路径
        local = torch.zeros(2, n_local, dtype=dtype, device="cuda")
        peer = self._share(local, group)
        pair = (local, peer)
        if len(self.staging) >= self.max_buffers:
            # 淘汰会丢弃对端仍映射的缓冲，两端必须淘汰同一 key；
            # 插入与访问顺序一致，OrderedDict 保证两端行为同步
            self.staging.popitem(last=False)
        self.staging[key] = pair
    return pair

def check_timeout(self) -> None:
    """在请求边界退休传输（设备读在 capture 内非法）。flag 由 spin 内核同时
    写本端与对端，因此超时会让两端一起退休；单端退休会导致 rank 发布
    对端永不发布的 collective，进而被 NCCL 看门狗杀进程。"""
    if self.failed or not self.inited or self.timed_out.item() == 0:
        return
    self.failed = True
    raise RuntimeError(
        "IPC all-to-all gave up waiting for its peer after "
        f"{envs.SGLANG_DIFFUSION_IPC_A2A_TIMEOUT_MS:g} ms, so that exchange "
        "returned incomplete data. The transport is now disabled on every "
        "rank; retry the request over NCCL."
    )
```

python/sglang/multimodal_gen/runtime/layers/usp.py

USPAttention 全家桶的接入点：新增四个 IPC 钩子并把 input/output/varlen A2A 入口改为先试 IPC 后回退 NCCL，覆盖 14 个模型。

```
def _ipc_input_a2a_qkv(q, k, v):
    """一次 attention 的三个输入 A2A 合并为一次 IPC 交换；不可用返回 None。
    三次交换共享同一 staging 槽位，只做一次 signal_and_wait 同步。"""
    if get_ulysses_parallel_world_size() != 2:
        return None
    group = _ipc_ready_group()
    if group is None:
        return None
    from sglang.multimodal_gen.runtime.distributed.device_communicators.ipc_a2a import (
        IPC_A2A,
    )

    b, s_local, h_global, d = q.shape
    half = h_global // 2
    r = IPC_A2A.rank
    n = b * s_local * half * d
```

```
    pair = IPC_A2A.get_staging(3 * n, 3 * n, q.dtype, group)
    if pair is None:
        return None
    local, peer = pair
    slot = IPC_A2A.next_slot()
    outs = []
    for i, t in enumerate((q, k, v)):
        # 把需要交给对端的 head 半区写入对端映射缓冲；本端保留自己的半区
        send = t[:, :, (1 - r) * half : (2 - r) * half].contiguous()
        peer[slot].narrow(0, i * n, n).copy_(send.view(-1), non_blocking=True)
        out = t.new_empty(b, 2 * s_local, half, d)
        out.narrow(1, r * s_local, s_local).copy_(t[:, :, r * half : (r + 1) * half])
        outs.append(out)
    # 一次同步覆盖 q/k/v 三个交换，替代 NCCL 的 3 次 rendezvous
    IPC_A2A.signal_and_wait()
    for i, out in enumerate(outs):
        theirs = local[slot].narrow(0, i * n, n).view(b, s_local, half, d)
        out.narrow(1, (1 - r) * s_local, s_local).copy_(theirs)
    return tuple(outs)
```

评论区精华

PR 讨论全部由作者 mickqian 在 CI 排障过程中沉淀，核心交锋如下：

- 默认值反复与双真值源：默认开启后 accuracy 失败，作者一度回退；随后发现 "The default was never actually off. envs resolves attributes through __getattr__ → the environment_variables dict, and that entry still read _lazy_bool("SGLANG_DIFFUSION_IPC_A2A", "true") while the dataclass stub said False"——stub 与 resolver 不一致导致 CI 一直在测试一个名为 opt-in 实为默认开的 PR。结论：统一真值来源到 resolver。
- 超时单端退休导致组失同步："A timeout retired the transport on one rank only, which desynchronises the group… the NCCL watchdog then takes the process down"——超时 rank 切到 NCCL 后发布的 collective 对端永远不发布。修复为 spin 内核同时写本端与对端 flag，check_timeout 在请求边界双端退休。
- B200 时钟源缺陷："cudaDevAttrClockRate… reports 1980 MHz on H100 (its peak) but 120 MHz on B200—so a 200 ms budget became roughly 13 ms of real time"——合法慢 peer 会被误判超时。改用 %globaltimer 纳秒墙钟，去掉 SM 时钟换算。
- 2-rank 门控的扩展性论证：独立微基准显示 "At 4 ranks the scheme is 25-45% slower"，成本在 R-1 次独立 copy_ 与 R-1 个串行轮询旗标两条轴上线性增长，而 NCCL 将多 peer 融合进单内核通道；R=2 时两者都收敛为单次远程写与单旗标，因此保持 2-rank 边界是刻意设计而非权宜。
- NCCL collective 无法录制进 CUDA graph："ProcessGroupNCCL bakes its per-op sequence number into the capture, so replayed kernels stop matching the peer's"——2xH100 上双 rank 卡死在 graph.replay()。IPC 交换是纯本地映射拷贝加 spin/bump 内核、无宿主往返与序列状态，天然 capture-safe，成为 #31852 全图捕获的使能前提。

- MI300 误报与测试哲学：ROCm 下传输必然不可用，两臂都走 NCCL，证据断言报 "IPC never engaged"——作者总结："a guard that proves an accelerated path executed needs a platform skip beside it, or it converts every other backend in the matrix into a false failure"。
- 默认开启后 accuracy 失败归因：zimage CosSim 0.9859、wan2_2_t2v_a14b 0.9882 低于阈值，一度归因存疑；最终定位为分支落后 main 108 个 commit 的 stale base（旧代码对刷新后的 ground truth 失配），rebase 后同一批 CI 全绿，与传输无关。
- envs 默认值双真值源：stub 与 resolver 不一致 (correctness): 统一真值来源：stub 与 resolver 同步修改，运行时以 resolver 为准。
- 超时单端退休导致进程组失同步 (correctness): spin 内核同时写本端与对端 flag，check_timeout 在请求边界双端退休并抛 RuntimeError 而非静默返回损坏数据。
- spin 超时时钟源：B200 上 cudaDevAttrClockRate 报 120 MHz (correctness): 改用 %globaltimer 纳秒墙钟，预算直接以纳秒传入，去掉 SM 时钟换算。
- 2-rank 门控的扩展性论证：R=4 时 IPC 输给 NCCL (performance): 保持 world_size == 2 门控是刻意设计；泛化实现正确但在 R=4 不划算，不落地。
- NCCL collective 无法安全录制进 CUDA graph，IPC 传输成为使能前提 (design): 本 PR 的 IPC 传输是 #31852 全前向图捕获在 multi-GPU 上可行的前提，这是除延迟收益外的第二价值。
- 默认开启后 accuracy 失败的归因：stale base 而非传输缺陷 (question): 失败与 IPC 传输无关，是 stale base；期间默认值一度回退再恢复，并补上了 AllToAll4D 的测试覆盖。
- MI300 误报与平台跳过：证据断言需要平台守卫配套 (testing): 按 current_platform.is_cuda() 跳过非 CUDA 平台；作者总结：证明加速路径执行过的断言必须搭配平台跳过，否则把其他后端变成假失败。

风险与影响

- 风险：技术风险集中在以下方面：
- 默认开启的影响面：最终提交将 SGLANG_DIFFUSION_IPC_A2A 默认置为开启，所有 2-rank + P2P 的 CUDA 扩散推理用户都会走新路径。虽有 parity 测试与逐位一致性保证，但真实负载中的 JIT 编译失败、P2P 探测差异、超时误判等失败路径行为仍需线上观察；envs.py 曾出现 stub/resolver 双真值源导致默认值漂移的教训。
- staging 缓存淘汰的一致性：get_staging 用 OrderedDict 先进先出淘汰，依赖两端以相同顺序插入与访问 key；任何一端新增 key 顺序不一致都会造成对端仍映射的缓冲被提前丢弃，属于分布式一致性的隐式契约。
- CUDA graph 捕获期间的退化：capture 内 staging Miss 返回 None 并回落 NCCL，若预热不充分，图内会烘焙 NCCL 路径——正确性保持但性能收益静默消失。
- 平台局限性：传输依赖 libcudart 与 cudaDeviceEnablePeerAccess，仅在 CUDA + P2P 可用；ROCm/NPU 下由测试跳过后台保障，但新增的 JIT 内核（ipc_a2a_sync）增加了启动时编译耗时与构建目录缓存依赖（SGLANG_DIFFUSION_CACHE_ROOT）。
- 并发拓扑覆盖不足：测试与实现仅覆盖纯 Ulysses 2-rank，未覆盖 ring + ulysses 组合或 varlen 与固定 shape 混跑的边界。

- 影响：
  - 用户侧：2×H100 ulysses=2 下 Qwen-Image-2512 1024² 从 81.3 ms/step 降到 64.8 ms/step（约 -20%，配合 #31852 全图降到 63-64.5 ms/step）；FLUX.1-dev 约 -8.4%；所有配置输出与 NCCL 路径 bitwise 一致。
  - 系统侧：新增设备通信抽象 IPC_A2A 与 JIT 内核依赖，gpu_worker.py 增加请求边界看门狗读取；USPAttention.forward 数据契约新增 qkv_pre_all_to_all 参数，涉及 14 个 USPAttention 模型与 3 个 stacked-qkv 模型（通过通用钩子，无需逐模型接线）。
  - 团队侧：新增 2-GPU parity 测试套件为后续通信改动提供回归保护，并沉淀了 " 证据断言 + 平台跳过 " 的测试方法论。
  - 影响边界：仅在 world_size == 2 且 head_dim == 2 时触发，其他拓扑与 rank 数完全不受影响。
  - 风险标记：默认开启影响面广，核心通信路径变更，分布式超时一致性，JIT 编译依赖，仅 CUDA + P2P 支持，staging 淘汰顺序隐式契约

关联脉络

- PR #31852 perf(diffusion): full-forward CUDA graph for the DiT (opt-in): 直接依赖与互证关系：本 PR 的 IPC 传输因 capture-safe 而成为 #31852 全前向 CUDA 图在 multi-GPU 上可运行的前提（NCCL collective 录制后不可回放），两者作为 PR 栈共同验证（PR body 提及的 #31849 未在历史列表中收录）。
- PR #31849 PR stack companion（历史列表未收录标题）：PR body 将 #31849 + #31852 + 本 PR 称为完整栈，并给出合并后 3.929-3.950s 的实测；历史 PR 列表未收录该 PR，具体内容无法核实。