

PR #38936 完整报告

sgl-project/sglang

[Fix] Disable NCCL graph buffer registration for the TP LM-head all-to-all (pure-DP decode hang under request bursts)

合并时间: 2026-09-12 04:10

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/38936>

执行摘要

- 一句话: 修复 TP LM-head all-to-all 纯 DP decode 死锁
- 推荐动作: 值得精读。这是一个典型的“实验矩阵定位 + 最小作用域 workaround + 参数层单测”的分布式死锁修复案例, 对部署 PD decode 或关注 CUDA graph 与 NCCL 交互的读者有参考价值。关注点: env 注入发生在 handle_data_parallelism 的时机、setdefault 保留用户显式设置的策略、以及作者如何用对照实验排除 MoE backend 与 dedicated buffers 两个变量。后续建议跟踪是否出现更根本的修复。

功能与动机

PR body 指出: Pure-DP PD decode nodes (tp_size == dp_size with DP attention) auto-enable the TP LM-head all-to-all, 一个 PyNccl send/recv captured in the decode CUDA graphs。With NCCL's default graph-capture buffer registration (NCCL_GRAPH_REGISTER=1) that exchange deadlocks the first time a burst of new requests ramps the DP ranks at different rates。作者在 tp4/dp4/ep4 unified launch 上确定性复现 (约 240 请求突发即挂起), 并在多节点 tp16/dp16/ep16 PD decode 部署上复现。实验矩阵显示 dedicated non-pool all-to-all buffers 单独无法解决, 而 NCCL_GRAPH_REGISTER=0 下 4/4、5/5、10/10 轮全部通过; NCCL 侧机制 (graph-pool temporaries 地址被池复用于其他张量导致注册交换失效) 为推断而非 NCCL 内部确认。

实现拆解

1. 变更入口: python/sglang/srt/arg_groups/parallel_hook.py 的 handle_data_parallelism 在依次执行 _tp_lm_head_all_to_all_default 与 _dp_lm_head_validation 两个 post-process pass 之后, 新增对 resolving_view(server_args).enable_tp_lm_head_all_to_all 的检查; 命中则调用新函数 _disable_nccl_graph_buffer_registration。选择在 handler 尾部注入, 是为了保证设置在 scheduler 创建 NCCL communicator (继承环境变量) 之前生效。
2. 核心逻辑: _disable_nccl_graph_buffer_registration 使用 os.environ.setdefault("NCCL_GRAPH_REGISTER", "0"), 仅在未显式设置时写入默认值; 若用户已显式设置非 "0" 值, 则保留原值并记录 WARNING, 提示该配置下 all-to-all 可能死锁。效果上, decode graph 内捕获的 LM-head all-to-all 与 DP gather 从注册零拷贝退化为 NCCL staging buffer, eager NCCL op 不受影响。

3. 作用域隔离：只有 `enable_tp_lm_head_all_to_all` 为真的路径会触发环境变量写入。
`attn_tp > 1` 的形状不会启用 all-to-all（该特性要求 `tp_size == dp_size`），
`--enable-dp-lm-head` 与 all-to-all 互斥，因此这两个配置的行为完全不变。
4. 测试配套：test/registered/unit/server_args/test_server_args.py 新增
`TestTpLmHeadAllToAllNcclGraphRegister`，覆盖三条路径——纯 DP decode 节点自动置 "0"、用户显式 `NCCL_GRAPH_REGISTER=1` 被保留且打警告、unified serving 与
`--enable-dp-lm-head` 下环境变量不被触碰。测试用 dummy model 与 patch.dict 隔离环境，属参数解析层单测；真实 2-GPU 的 `test_tp_lm_head_all_to_all_cuda_graph` 测试留给 CI。
5. 验证与部署：作者在内部 tp4/dp4/ep4 环境跑 10 轮突发（240-256 并发、250-1000 输入 token、20-400 输出 token）全部通过且无 watchdog；tp4/dp2/ep4 确认环境变量不被改动，即无行为变化。

关键文件：

- python/sglang/srt/arg_groups/parallel_hook.py（模块 参数解析；类别 source；类型 core-logic；符号 `handle_data_parallelism`, `_disable_nccl_graph_buffer_registration`）：核心修复所在：`handle_data_parallelism` 在 TP LM-head all-to-all 解析为开启时调用新函数 `_disable_nccl_graph_buffer_registration`，通过 `os.environ.setdefault` 把 `NCCL_GRAPH_REGISTER` 置为 0，并保留用户显式设置且记录 WARNING；这是消除纯 DP decode 突发死锁的关键路径。
- test/registered/unit/server_args/test_server_args.py（模块 参数测试；类别 test；类型 test-coverage；符号 `TestTpLmHeadAllToAllNcclGraphRegister`, `_resolve`, `test_pure_dp_decode_node_disables_nccl_graph_register`, `test_explicit_nccl_graph_register_is_kept`）：新增
`TestTpLmHeadAllToAllNcclGraphRegister` 三个用例，验证默认置 0、用户显式值保留并告警、非 all-to-all 路径不动环境变量，构成参数解析层的回归保障。

关键符号：`_disable_nccl_graph_buffer_registration`, `handle_data_parallelism`,
`TestTpLmHeadAllToAllNcclGraphRegister._resolve`,
`test_pure_dp_decode_node_disables_nccl_graph_register`,
`test_explicit_nccl_graph_register_is_kept`, `test_without_all_to_all_env_is_untouched`

关键源码片段

python/sglang/srt/arg_groups/parallel_hook.py

核心修复所在：`handle_data_parallelism` 在 TP LM-head all-to-all 解析为开启时调用新函数 `_disable_nccl_graph_buffer_registration`，通过 `os.environ.setdefault` 把 `NCCL_GRAPH_REGISTER` 置为 0，并保留用户显式设置且记录 WARNING；这是消除纯 DP decode 突发死锁的关键路径。

```
# python/sglang/srt/arg_groups/parallel_hook.py（本次变更整理）
def handle_data_parallelism(server_args: Any):
    # 前面还有 prefill CUDA graph max_bs 的 DP 钳制逻辑，此处从
    # LM-head 默认值解析开始：先解析 phase-aware TP LM-head 默认值，
    # 再校验 DP/TP LM-head 配置合法性。
```

```
run_post_process_pass(server_args, _tp_lm_head_all_to_all_default)
run_post_process_pass(server_args, _dp_lm_head_validation)
```

```
# 只要 TP LM-head all-to-all 最终解析为开启，就必须在 scheduler
# 创建 NCCL communicator 之前关闭图缓冲注册，否则请求突发时
# DP 各 rank 会在 ncclDevKernel_SendRecv 内互相等待从而死锁。
if resolving_view(server_args).enable_tp_lm_head_all_to_all:
    _disable_nccl_graph_buffer_registration()
```

```
def _disable_nccl_graph_buffer_registration() -> None:
```

```
    """让图捕获的 PyNccl all-to-all 走 NCCL staging buffer 而非注册零拷贝。
```

```
    NCCL_GRAPH_REGISTER（默认开启）会把 CUDA graph 内捕获的每个 collective
    的 send/recv 缓冲区在图生命周期内注册，peer 直接通过注册表搬运数据。
    TP LM-head all-to-all 被捕获在 decode graph 的 graph-pool 临时张量上，
    而这些地址会被 pool 复用于其他张量，注册交换无法在这种场景下存活：
    突发请求使 DP rank 以不同速率爬升时，一个 rank 已离开当前 step，其余
    rank 永远旋转在 ncclDevKernel_SendRecv 中，最终看门狗触发整组重启。
    必须在 scheduler 创建继承该环境的 NCCL communicator 之前执行；
    用户显式设置的值优先保留。
```

```
    """
```

```
    if os.environ.setdefault("NCCL_GRAPH_REGISTER", "0") != "0":
        logger.warning(
            "NCCL_GRAPH_REGISTER=%s was set explicitly; the graph-captured TP "
            "LM-head all-to-all can deadlock with registered buffers.",
            os.environ["NCCL_GRAPH_REGISTER"],
        )
```

test/registered/unit/server_args/test_server_args.py

新增 TestTpLmHeadAllToAllNcclGraphRegister 三个用例，验证默认置 0、用户显式值保留并告警、非 all-to-all 路径不动环境变量，构成参数解析层的回归保障。

```
# test/registered/unit/server_args/test_server_args.py（新增测试类整理）
class TestTpLmHeadAllToAllNcclGraphRegister(unittest.TestCase):
    """图捕获的 TP LM-head all-to-all 不能与 NCCL graph buffer registration
    同时启用：注册过的 graph-pool 临时张量会在 DP rank 速率不一致时死锁。"""

    def _resolve(self, **kwargs):
        # 用 dummy model 构造 tp2/dp2 + DP attention 的 decode 配置，
        # 让参数解析管线走到 handle_data_parallelism 后直接返回。
        server_args = ServerArgs(
            model_path="dummy",
            enable_dp_attention=True,
            tp_size=2,
            dp_size=2,
            chunked_prefill_size=8192,
            cuda_graph_config=CudaGraphConfig(
                prefill=PhaseConfig(backend=Backend.DISABLED)
```

```

    ),
    **kwargs,
)
parallel_hook.handle_data_parallelism(server_args)
return server_args

def test_pure_dp_decode_node_disables_nccl_graph_register(self):
    # 纯 DP decode 节点应自动把 NCCL_GRAPH_REGISTER 置为 "0"。
    with patch.dict(os.environ, {}, clear=False):
        os.environ.pop("NCCL_GRAPH_REGISTER", None)
        server_args = self._resolve(disaggregation_mode="decode")
        self.assertTrue(
            resolution_result(server_args, "enable_tp_lm_head_all_to_all")
        )
        self.assertEqual(os.environ.get("NCCL_GRAPH_REGISTER"), "0")

def test_explicit_nccl_graph_register_is_kept(self):
    # 用户显式设置的值要保留，但要发出 WARNING 提示死锁风险。
    with patch.dict(os.environ, {"NCCL_GRAPH_REGISTER": "1"}, clear=False):
        with self.assertLogs(parallel_hook.logger, level="WARNING") as logs:
            self._resolve(disaggregation_mode="decode")
        self.assertEqual(os.environ["NCCL_GRAPH_REGISTER"], "1")
        self.assertIn("NCCL_GRAPH_REGISTER=1", "\n".join(logs.output))

def test_without_all_to_all_env_is_untouched(self):
    # unified serving 默认不开 all-to-all，decode 节点配合
    # --enable-dp-lm-head 也不会走 all-to-all，这两种情况都不动环境变量。
    for kwargs in (
        {},
        {"disaggregation_mode": "decode", "enable_dp_lm_head": True},
    ):
        with self.subTest(**kwargs), patch.dict(os.environ, {}, clear=False):
            os.environ.pop("NCCL_GRAPH_REGISTER", None)
            server_args = self._resolve(**kwargs)
            self.assertFalse(
                resolution_result(server_args, "enable_tp_lm_head_all_to_all")
            )
            self.assertNotIn("NCCL_GRAPH_REGISTER", os.environ)

```

评论区精华

该 PR 没有正式 review 评论，讨论集中在 PR body 的实验矩阵和作者的 Issue 评论。作者两次使用 /tag-and-rerun-ci 触发 CI 重跑，最后一次评论说明 "The only failing test is unrelated, merging."。也就是说，合并决策主要建立在作者自证的对照实验而非 reviewer 交锋之上，这是阅读该 PR 时需要注意的一点。

- CI 失败与合并决策 (other): 以作者对失败测试的判定为准合并，未进一步核实失败与本变更的关联。

风险与影响

- 风险:

1. 进程级环境变量副作用: `NCCL_GRAPH_REGISTER=0` 一旦写入, 会影响同一 scheduler 进程内所有后续创建的 NCCL communicator 的图捕获行为。除了 LM-head all-to-all, 图中其他被捕获的 collectives (如 MoE all-to-all) 也会从注册零拷贝退化为 staging buffer, 可能带来图内通信性能损耗, PR 中未提供该路径的吞吐对比。
2. 机制推断的不确定性: PR body 明确说明死锁的 NCCL 侧机制是 *inferred from the experiment matrix, not from NCCL internals*。NCCL 版本升级或 CUDA graph pool 地址复用策略变化, 可能导致死锁以其他形式复现。
3. 显式配置的漏网: 用户显式设置 `NCCL_GRAPH_REGISTER=1` 时只记录 WARNING 不强制覆盖, 死锁风险依然存在。
4. 时序依赖: `env` 注入依赖 `handle_data_parallelism` 在 communicator 创建前被调用的管线顺序; 若未来参数解析顺序调整, 该设置可能失效。
5. 测试与 CI 缺口: 单测仅覆盖参数解析层, 无真实多卡压力回归; CI 的 AMD ROCm 10 job 失败被作者判定为无关, 但没有附上排查证据。
 - 影响: 用户侧: 修复 pure-DP PD decode 部署在请求突发时的全局挂起与看门狗重启, 直接提升多节点 PD decode 稳定性, 尤其利好 `tp_size == dp_size` 且启用 DP attention 的部署。系统侧: scheduler 进程新增一个环境变量, decode graph 内捕获的 NCCL collective 从注册零拷贝退化为 staging buffer, 仅影响 all-to-all 解析为开启的进程。团队侧: 为并行参数解析新增了环境变量注入模式, 可复用于类似 NCCL/ 图捕获兼容性问题; 但该修复属于 workaround, 后续应跟踪是否有更底层的 NCCL graph mode 或专用 buffer 方案。
 - 风险标记: 进程级环境变量副作用, 机制基于推断, 显式 `NCCL_GRAPH_REGISTER=1` 仍可能死锁, 缺少真实多卡回归, CI 失败未核实关联

关联脉络

- PR #38993 [Refactor] Generalize attention graph variants in the decode runner: 同为 decode CUDA graph 路径的注意力图变体重构, 后续对 graph capture 行为的调整需与本次 all-to-all 环境变量注入协同考虑。
- PR #36612 [PD] Share the prefill->decode failure notification across backends: 同属 PD decode 稳定性与失败处理方向, 体现 PD 生命周期在多后端上的持续加固。
- PR #38756 [AMD] aiter: resolve SWA KV pool for draft workers + guard paged decode: 同为 decode 路径上的守卫类 bugfix, 构成 DP/PD decode 稳定性修复系列, 便于追溯同域修复脉络。