PR #29211 完整报告

sgl-project/sglang

[disagg] Fix KV-event publisher port collision under pure data parallelism

合并时间：2026-07-02 01:57

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/29211

执行摘要

- 一句话：修复纯数据并行下 KV-event 端口冲突
- 推荐动作：建议阅读此 PR，特别是 select_kv_publisher_dp_rank 函数的设计——它简洁地区分了纯 DP 和 DP-attention 两种模式，利用了已有的 attn_dp_size 参数而无需引入新配置。对于从事解聚或多数据中心并行开发的工程师有价值。

功能与动机

当引擎运行纯数据并行（--dp-size N 而不启用 --enable-dp-attention）并启用 KV 事件发布时，所有 N 个数据并行工作者都在同一个 ZMQ 端口上发布 KV 事件。因为每个工作者的 attn_dp_rank 都是 0，所以端口偏移量相同，导致绑定端口冲突（binding endpoint）或订阅端接收不到部分数据（connecting endpoint）。

实现拆解

1. 在 python/sglang/srt/disaggregation/kv_events.py 中新增 select_kv_publisher_dp_rank 函数。该函数接收 attn_dp_size、attn_dp_rank 和 dp_rank，当 attn_dp_size > 1（DP-attention 模式）时返回 attn_dp_rank，否则（纯 DP 或无 DP）返回 dp_rank（若 dp_rank 为 None 则返回 0）。
2. 在 python/sglang/srt/managers/scheduler_components/kv_events_publisher.py 中导入 select_kv_publisher_dp_rank，并在 init_kv_events 方法中调用它替换原来直接使用 self.ps.attn_dp_rank 作为 EventPublisherFactory.create 的 rank 参数。
3. 新增 test/registered/unit/disaggregation/test_kv_events.py 单元测试，注册为 CPU 测试（base-a-test-cpu）。测试内容包括：select_kv_publisher_dp_rank 在各种模式下的返回值（test_select_rank_across_modes）、模拟端口偏移确保顺序端口分配（test_workers_bind_sequential_ports_per_replica）、以及验证发布者排名数量与 /server_info 中 dp_size 一致（test_publisher_rank_count_matches_advertised_dp_size）。

关键文件：

- python/sglang/srt/disaggregation/kv_events.py（模块 解聚；类别 source；类型 core-logic；符号 select_kv_publisher_dp_rank）：核心逻辑：新增 select_kv_publisher_dp_rank 函数，根据并行模式选择正确的端口偏移排名。

- python/sglang/srt/managers/scheduler_components/kv_events_publisher.py（模块 解聚；类别 source；类型 core-logic）：调用点：在 init_kv_events 中使用新增函数替换原始 attn_dp_rank，确保端口偏移正确。
- test/registered/unit/disaggregation/test_kv_events.py（模块 解聚；类别 test；类型 test-coverage；符号 TestSelectKvPublisherDpRank, test_select_rank_across_modes, test_workers_bind_sequential_ports_per_replica, test_publisher_rank_count_matches_advertised_dp_size）：新增单元测试覆盖 select_kv_publisher_dp_rank 在纯 DP、DP-attention 和单副本模式下的行为，以及端口顺序和数量验证，确保修复正确性。

关键符号：select_kv_publisher_dp_rank, SchedulerKvEventsPublisher.init_kv_events, EventPublisherFactory.create

关键源码片段

test/registered/unit/disaggregation/test_kv_events.py

新增单元测试覆盖 select_kv_publisher_dp_rank 在纯 DP、DP-attention 和单副本模式下的行为，以及端口顺序和数量验证，确保修复正确性。

```
import unittest
from sglang.srt.disaggregation.kv_events import (
    ZmqEventPublisher,
    select_kv_publisher_dp_rank,
)
from sglang.test.ci.ci_register import register_cpu_ci
from sglang.test.test_utils import CustomTestCase

register_cpu_ci(est_time=2, suite="base-a-test-cpu")


class TestSelectKvPublisherDpRank(CustomTestCase):
    def test_select_rank_across_modes(self):
        """验证 select_kv_publisher_dp_rank 在不同模式下的返回值"""
        # (label, attn_dp_size, attn_dp_rank, dp_rank, expected)
        cases = [
            # 纯 DP：用 dp_rank 区分
            ("pure_dp_worker0", 1, 0, 0, 0),
            ("pure_dp_worker1", 1, 0, 1, 1),
            ("pure_dp_worker3", 1, 0, 3, 3),
            # DP-attention：用 attn_dp_rank 区分，忽略 dp_rank
            ("dp_attention_rank0", 2, 0, None, 0),
            ("dp_attention_rank1", 2, 1, None, 1),
            ("dp_attention_ignores_dp_rank", 2, 1, 99, 1),
            # 单副本：dp_rank 为 None 或 0 都返回 0
            ("single_dp_rank_none", 1, 0, None, 0),
            ("single_dp_rank_zero", 1, 0, 0, 0),
        ]
        for label, attn_dp_size, attn_dp_rank, dp_rank, expected in cases:
```

```
        with self.subTest(label):
            self.assertEqual(
                select_kv_publisher_dp_rank(attn_dp_size, attn_dp_rank, dp_rank),
                expected,
            )

    def test_workers_bind_sequential_ports_per_replica(self):
        """验证顺序端口映射：每个副本 r 应绑定 port_base + r"""
        endpoint = "tcp://*:5557"
        expected = [f"tcp://*:{5557 + r}" for r in range(4)]

        # 纯 DP：副本索引为 dp_rank
        pure_dp = [
            ZmqEventPublisher.offset_endpoint_port(
                endpoint, select_kv_publisher_dp_rank(1, 0, r)
            )
            for r in range(4)
        ]
        self.assertEqual(pure_dp, expected)

        # DP-attention：副本索引为 attn_dp_rank
        dp_attention = [
            ZmqEventPublisher.offset_endpoint_port(
                endpoint, select_kv_publisher_dp_rank(4, a, None)
            )
            for a in range(4)
        ]
        self.assertEqual(dp_attention, expected)

    def test_publisher_rank_count_matches_advertised_dp_size(self):
        """验证发布者排名数量与 dp_size 一致，保证订阅端能收到完整数据"""
        for dp_size in (1, 2, 4):
            with self.subTest(f"pure_dp_{dp_size}"):
                ranks = {
                    select_kv_publisher_dp_rank(
                        attn_dp_size=1, attn_dp_rank=0, dp_rank=r
                    )
                    for r in range(dp_size)
                }
                self.assertEqual(len(ranks), dp_size)
            with self.subTest(f"dp_attention_{dp_size}"):
                ranks = {
                    select_kv_publisher_dp_rank(
                        attn_dp_size=dp_size, attn_dp_rank=a, dp_rank=None
                    )
                    for a in range(dp_size)
                }
                self.assertEqual(len(ranks), dp_size)
```

评论区精华

审核者 ShangmingCai 批准了 PR，并 *cc* @ishandhanani。无其他评论或讨论。

- 暂无高价值评论线程

风险与影响

- 风险：变更集中在 KV 事件发布端口映射逻辑，影响范围限定于启用了 KV 事件发布的解聚引擎。核心风险是：如果 *attn_dp_size* 和 *dp_rank* 意外组合导致返回错误排名，可能引发端口冲突或数据缺失。但新增的单元测试覆盖了主要组合，且逻辑简单（分支仅依赖 *attn_dp_size* > 1 判断），风险较低。另外，需注意对 DP-attention 模式的兼容性：本变更仅改变纯 DP 模式的行为，DP-attention 模式行为不变。
- 影响：对用户影响：修复了在纯 DP 模式下启用 KV 事件发布时引擎启动崩溃或订阅数据不完整的 bug。对系统影响：端口分配从原来的始终使用 *attn_dp_rank*（恒为 0）变为根据模式选择，确保纯 DP 模式下端口正确偏移。对团队影响：提供了清晰的代码分离和测试用例，便于后续维护。
- 风险标记：端口映射逻辑变更，纯 DP 场景修复，单元测试覆盖

关联脉络

- 暂无明显关联 PR