PR #22983 完整报告

sgl-project/sglang

[KV-Events] Fix kv events events publishing for CP

合并时间：2026-04-20 17:34

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/22983

执行摘要

- 一句话：修复上下文并行下 KV 事件发布的重复问题，确保每个数据并行 rank 只发布一个事件流。
- 推荐动作：建议工程师精读此 PR，特别是对于涉及分布式 rank 管理和事件发布的代码，关注如何优雅处理 CP 和 TP 的协同，并参考测试用例验证类似场景。

功能与动机

根据 PR body，在上下文并行中，多个 CP rank 可能都有 attn_tp_rank == 0，导致每个 CP rank 尝试为同一个 DP rank 发布 KV 事件，造成 ZMQ 重复发布者和绑定冲突（如 tcp://*:5557），语义上也会复制 KV 缓存事件。

实现拆解

1. 修改调度器事件初始化逻辑：在 python/sglang/srt/observability/scheduler_metrics_mixin.py 的 init_kv_events 方法中，将条件从 kv_events_config and self.attn_tp_rank == 0 改为 kv_events_config and self.attn_tp_rank == 0 and self.attn_cp_rank == 0，确保只有 TP 和 CP rank 都为 0 时才启用事件发布。
2. 添加回归测试：在 test/manual/test_kv_events.py 中添加新测试函数 test_kv_events_attn_cp_single_stream_per_dp_rank，使用 Qwen 模型配置 TP2 和 attn-cp-size 2，验证 CP replicas 不发布重复事件。
3. 测试覆盖验证：测试检查事件流是否仅从 attn_dp_rank=0 发布，并确保没有意外事件流。

关键文件：

- python/sglang/srt/observability/scheduler_metrics_mixin.py（模块 调度器观测；类别 source；类型 core-logic；符号 init_kv_events）：核心逻辑变更处，修改了 KV 事件发布的条件，直接影响事件流生成
- test/manual/test_kv_events.py（模块 KV 事件测试；类别 test；类型 test-coverage；符号 test_kv_events_attn_cp_single_stream_per_dp_rank）：添加回归测试验证修复，确保 CP 下事件流唯一性

关键符号：init_kv_events, test_kv_events_attn_cp_single_stream_per_dp_rank

关键源码片段

python/sglang/srt/observability/scheduler_metrics_mixin.py

核心逻辑变更处，修改了 KV 事件发布的条件，直接影响事件流生成

```
def init_kv_events(self: Scheduler, kv_events_config: Optional[str]):
    # 变更前：仅检查 attn_tp_rank == 0
    # 变更后：增加 attn_cp_rank == 0 条件，确保在上下文并行下只有第一个 CP rank 发布事件
    # 这匹配了 TP 的行为，并防止多个 CP replicas 为同一个 DP rank 发布重复事件
    self.enable_kv_cache_events = bool(
        kv_events_config and self.attn_tp_rank == 0 and self.attn_cp_rank == 0
    )

    if self.enable_kv_cache_events:
        self.kv_event_publisher = EventPublisherFactory.create(
            kv_events_config, self.attn_dp_rank
        )
```

test/manual/test_kv_events.py

添加回归测试验证修复，确保 CP 下事件流唯一性

```
def test_kv_events_attn_cp_single_stream_per_dp_rank(self):
    """Test that CP replicas do not publish duplicate KV events for one DP rank."""
    # 设置 ZMQ 订阅器，一个用于预期事件流（端口 5557），另一个用于检测意外流（端口 5558）
    sub_dp0 = context.socket(zmq.SUB)
    sub_dp0.connect("tcp://localhost:5557")
    topic = "kv-events"
    sub_dp0.setsockopt_string(zmq.SUBSCRIBE, topic)

    sub_unexpected = context.socket(zmq.SUB)
    sub_unexpected.connect("tcp://localhost:5558")
    sub_unexpected.setsockopt_string(zmq.SUBSCRIBE, topic)

    # 启动服务器，配置 TP 和 CP 以模拟上下文并行场景
    process = popen_launch_server(
        QWEN3_30B_MODEL_PATH,
        DEFAULT_URL_FOR_TEST,
        timeout=DEFAULT_TIMEOUT_FOR_SERVER_LAUNCH,
        other_args=[
            "--kv-events-config", '{"publisher": "zmq", "topic": "kv-events"}',
            "--tp-size", 2, # 张量并行大小为 2
            "--attn-cp-size", 2, # 注意力上下文并行大小为 2
            "--moe-dp-size", 2,
            "--enable-prefill-context-parallel",
            "--trust-remote-code",
            # ... 其他参数省略以保持简洁
        ],
    )

    try:
        # 发送请求以生成 KV 事件
```

```
    for i in range(4):
        response = requests.post(
            f"{DEFAULT_URL_FOR_TEST}/generate",
            json={
                "text": f"KV event context parallelism request {i}: write a concise fact about
                distributed inference.",
                "sampling_params": {"temperature": 0, "max_new_tokens": 16},
            },
        )
        self.assertEqual(response.status_code, 200)

    # 验证事件批次：检查 attn_dp_rank 是否为 0，并确保没有意外事件流
    batches = []
    stored_hashes = set()
    duplicate_hashes = set()
    unexpected_batches = []
    start = time.time()
    max_wait_s = 15
    min_stored_blocks = 3

    while (time.time() - start) < max_wait_s and (len(stored_hashes) < min_stored_blocks):
        if sub_dp0.poll(timeout=100):
            _, seq_bytes, payload = sub_dp0.recv_multipart()
            event_batch = decoder.decode(payload)
            self.assertEqual(
                event_batch.attn_dp_rank, 0,
                "CP mode with one DP rank should publish events as attn_dp_rank=0",
            )
            batches.append(event_batch)
            # 进一步验证事件类型和唯一性
            for event in event_batch.events:
                self.assertIsInstance(event, (BlockStored, BlockRemoved, AllBlocksCleared))
                if isinstance(event, BlockStored):
                    for block_hash in event.block_hashes:
                        if block_hash in stored_hashes:
                            duplicate_hashes.add(block_hash)
                        stored_hashes.add(block_hash)

        if sub_unexpected.poll(timeout=0):
            self.fail("Unexpected event stream detected from CP replica")

    # 断言确保没有重复哈希和意外批次
    self.assertEqual(len(duplicate_hashes), 0, "Duplicate block hashes detected")
    self.assertEqual(len(unexpected_batches), 0, "Unexpected event batches from CP
    replicas")
finally:
    # 清理资源
    sub_dp0.close()
    sub_unexpected.close()
```

```
    context.term()
    kill_process_tree(process.pid)
```

评论区精华

Review 中只有简短批准，无深入讨论。hzh0425 和 ShangmingCai 均 APPROVED，结论是修复正确且 CI 超时无关。

- 修复批准与合并 (other): 修复被接受并合并，CI 超时不影响代码正确性。

风险与影响

- 风险：主要风险是条件逻辑可能过于严格，如果未来架构变更（如 rank 计算调整）可能导致事件丢失。但测试覆盖了特定场景（TP2+attn-cp-size 2），降低了回归风险。此外，事件发布依赖于 rank 条件，若初始化逻辑错误可能影响观测性。
- 影响：修复了 CP 下 KV 事件重复发布的问题，避免了 ZMQ 绑定冲突和事件冗余，提升了观测性工具的可靠性。对使用上下文并行的用户，事件流将更准确，有助于调试和监控分布式推理状态。
- 风险标记：事件流冲突，条件逻辑风险

关联脉络

- PR #21249 Support allreduce fusion with cp: 同样涉及上下文并行（CP）的处理，本 PR 修复了 CP 下的事件发布问题，可视为 CP 功能演进的一部分。