PR #46913 完整报告

vllm-project/vllm

[communication] [bugfix] fix quickreduce acc error in cudagraph mode

合并时间：2026-07-27 16:38

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/46913

执行摘要

- 一句话：修复 CUDAGraph 模式下 QuickReduce 的 flag 冻结错误
- 推荐动作：建议精读此 PR，尤其是设备端计数器模式的设计。它展示了如何绕过 CUDA 图固定输入的限制，通过设备端持久化状态变量来保持动态变化。测试用例的多进程 + CUDA 图隔离写法也值得学习。

功能与动机

在 CUDA 图模式下，flag_color 被图固定后每轮保持不变，写入的 flag 值与上一轮残留值无法区分，导致等待方被过早满足而读取未就绪的数据，造成计算结果错误。PR body 通过验证结果展示了修复前 round 1 本应得到 4.0 却得到 1.0 的典型错误。

实现拆解

实现拆解

步骤 1：修改 CUDA kernel 签名与逻辑 在 allreduce_prototype_twoshot 内核中将参数 uint32_t flag_color 替换为 uint32_t* d_flag_counters。内核开始处从设备内存加载当前块的计数器值作为 flag_color，在内核循环结束后，由 block 的 (0,0) 线程将最终 flag_color 写回 d_flag_counters[blockIdx.x]，实现计数器在设备端的自增。

步骤 2：更新所有 kernel launch 宏 在 TWOSHOT_DISPATCH 和 TWOSHOT_DISPATCH_TP2_ONLY 宏中，将 kernel launch 调用从传递 flag_color 改为传递 d_flag_counters 指针。

步骤 3：修改 DeviceComms 结构体 将成员 uint32_t flag_color 替换为 uint32_t* d_flag_counters，在 Initialize 函数中为 d_flag_counters 分配设备内存并初始化为 1（与原有 flag_color 初值一致），在 Destroy 函数中释放该设备内存。移除主机端每轮递增 flag_color 的代码，因为现在颜色旋转在内核中自动完成。

步骤 4：添加 CUDAGraph 重放测试 在 tests/distributed/test_rocm_quick_reduce.py 中新增 _quick_allreduce_cudagraph_worker 和 test_quick_allreduce_cudagraph_replay。该测试启动 2 个进程，每个进程构建一个仅包含单次 quickreduce 调用的 CUDA 图，然后重复重放 10 轮，检查每轮 all-reduce 结果是否等于 world_size * v。采用 multi_gpu_test 装饰器实现多 GPU 并行测试。

关键文件：

- csrc/quickreduce/quick_reduce.h（模块 快速归约；类别 source；类型 core-logic；符号 allreduce_prototype_twoshot, DeviceComms, TWOSHOT_DISPATCH, TWOSHOT_DISPATCH_TP2_ONLY）：核心逻辑变更：修改内核 allreduce_prototype_twoshot 和结构体 DeviceComms，用设备端计数器数组解决 CUDA 图重放时的 flag 冻结问题。
- tests/distributed/test_rocm_quick_reduce.py（模块 测试；类别 test；类型 test-coverage；符号 _quick_allreduce_cudagraph_worker, expected, _run_cudagraph_replay_test, test_quick_allreduce_cudagraph_replay）：新增 CUDAGraph 重放测试，验证修复后多轮 all-reduce 结果正确。

关键符号：allreduce_prototype_twoshot, DeviceComms::Initialize, DeviceComms::Destroy, _quick_allreduce_cudagraph_worker, _run_cudagraph_replay_test, test_quick_allreduce_cudagraph_replay

关键源码片段

csrc/quickreduce/quick_reduce.h

核心逻辑变更：修改内核 allreduce_prototype_twoshot 和结构体 DeviceComms，用设备端计数器数组解决 CUDA 图重放时的 flag 冻结问题。

```
// 修改后的内核：从设备内存读取每个块的 flag_counter，并在结束时写回
__global__ __quickreduce_launch_bounds_two_shot__ static void
allreduce_prototype_twoshot(T const* A, T* B, uint32_t N, uint32_t num_blocks,
                            int rank, uint8_t** dbuffer_list,
                            uint32_t data_offset, uint32_t* d_flag_counters,
                            int64_t data_size_per_phase) {
  int block = blockIdx.x;
  int grid = gridDim.x;

  // 从设备内存加载当前块的 flag_color，绕过 CUDA 图对参数常量的冻结
  uint32_t flag_color = d_flag_counters[blockIdx.x];

  while (block < num_blocks) {
    AllReduceKernel::run(A, B, N, block, rank, dbuffer_list, data_offset,
                         flag_color, data_size_per_phase);
    block += grid;
    flag_color++; // 每个 grid 循环增加一次颜色
  }
  // 只有线程 (0,0) 写回，避免同一 block 内多线程竞争
  if (threadIdx.x == 0 && threadIdx.y == 0) {
    d_flag_counters[blockIdx.x] = flag_color;
  }
}

// 结构体变更：用设备指针替换标量颜色，并在初始化时分配设备内存
struct DeviceComms {
  // ... 其他成员
  uint32_t* d_flag_counters = nullptr; // 替换原 uint32_t flag_color = 1
```

```
  void Initialize(...) {
    // 在函数内分配设备内存并初始化为 1
    HIP_CHECK(hipMalloc(&d_flag_counters, kMaxWorldSize * kMaxNumBlocks * sizeof(uint32_t))
    );
    HIP_CHECK(hipMemset(d_flag_counters, 0, kMaxWorldSize * kMaxNumBlocks * sizeof(uint32_
    t)));
    // ... 其他初始化
  }

  void Destroy() {
    if (d_flag_counters) hipFree(d_flag_counters);
    // ...
  }
};
```

tests/distributed/test_rocm_quick_reduce.py

新增 CUDAGraph 重放测试，验证修复后多轮 all-reduce 结果正确。

```
# 辅助函数：每个 worker 进程执行一次 quickreduce 的 CUDA 图重放测试
def _quick_allreduce_cudagraph_worker(
    rank: int, world_size: int, port: int, quant_level: str
):
    # 设置环境变量确保量化模式一致
    os.environ["VLLM_ROCM_QUICK_REDUCE_QUANTIZATION"] = quant_level
    os.environ["VLLM_ROCM_QUICK_REDUCE_CAST_BF16_TO_FP16"] = "0"
    device = torch.device(f"cuda:{rank}")
    torch.accelerator.set_device_index(device)
    dist.init_process_group(backend="gloo", init_method=f"tcp://127.0.0.1:{port}",
                            rank=rank, world_size=world_size)
    qar = QuickAllReduce(group=dist.GroupMember.WORLD, device=rank)
    N = 1 << 21 # 2M fp16 = 4 MB
    inp = torch.empty(N, dtype=torch.float16, device=device)
    out = torch.empty(N, dtype=torch.float16, device=device)
    assert qar.should_quick_allreduce(inp)

    def expected(v):
        return float(world_size * v)

    # 预热并记录 CUDA 图，图中仅包含一次 quickreduce 调用
    inp.fill_(1.0)
    qar.quick_all_reduce(inp, out=out)
    torch.accelerator.synchronize()
    dist.barrier()
    g = torch.cuda.CUDAGraph()
    with torch.cuda.graph(g):
        qar.quick_all_reduce(inp, out=out)
    torch.accelerator.synchronize()
    dist.barrier()

    for v in range(10):
```

```
            inp.fill_(float(v)) # 每轮设置相同值
            dist.barrier()
            g.replay()
            torch.accelerator.synchronize()
            dist.barrier()
            got = out.float()
            expect = expected(v)
            mismatch = ~torch.isclose(got, torch.full_like(got, expect))
            assert int(mismatch.sum()) == 0, (
                f"rank={rank} round={v} mismatched {mismatch.sum()}/{got.numel()}"
            )
        qar.close()

    # 测试入口，启动多进程运行 worker
    @multi_gpu_test(num_gpus=CUDAGRAPH_WORLD_SIZE)
    @torch.inference_mode()
    def test_quick_allreduce_cudagraph_replay(quant_level: str):
        _run_cudagraph_replay_test(quant_level)
```

评论区精华

- ilmarkov 的注释建议：在 csrc/quickreduce/quick_reduce.h 中，ilmarkov 指出 "Color rotation happens inside the kernel now; nothing to do on the host." 注释可以移除，因为过于明显或多余。从最终 patch 看，该注释已被简化保留。
- tjtanaa 指出 GPU 数量字符串错误：在测试文件中，用于跳过条件的字符串写成了 "requires 4 ROCm GPUs"，但实际 CUDAGRAPH_WORLD_SIZE 为 2，应为 "2"。该问题在后续提交中已修正。
- tjtanaa 建议将测试添加到单元测试脚本：在 issue 评论中，tjtanaa 请求将 cudagraph 测试用例添加到现有的 tests/distributed/test_rocm_quick_reduce.py 中，该请求已在 PR 中直接实现。
- 移除多余的注释 (style): 注释被简化为 `// Color rotation happens inside the kernel now; nothing to do on the host.`，保留但更简洁。
- 测试跳过条件字符串错误 (testing): 后续提交修正为 "requires 2 ROCm GPUs"。
- 建议将测试添加到单元测试文件 (testing): PR 中已直接在对应文件中新增测试，满足要求。

风险与影响

- 风险：
  - 设备端计数器初始化缺失：若 d_flag_counters 未正确分配或初始化为 1，可能导致标志颜色错误，造成竞态条件。需确保初始化与销毁配对。
  - 仅覆盖 2 GPU 场景：新增的 cudagraph 测试仅在 2 GPU 环境下运行，未覆盖 4/8 GPU 场景，可能存在未发现的 bug。
  - 多进程测试稳定性：测试使用 multiprocessing 启动子进程，在 CI 环境可能因资源争抢或超时而偶发失败。

    - 内核修改影响所有量化级别：allreduce_prototype_twoshot 内核被所有量化级别（FP/INT8/INT6/INT4/INT3）共用，修改后需确保各量化级别在 cudagraph 模式下均正确。
- 影响：
    - 用户：修复了 ROCm 平台上使用 CUDA 图时的 QuickReduce 精度错误，用户不再因该 bug 得到错误的推理结果。
    - 系统：无性能回归（PR 声称），但增加了设备端内存开销（每个块 4 字节），对典型模型可以忽略。
    - 团队：为分布式通信组件提供了一种处理图冻结标志位的标准模式，可供其他类似组件参考。
    - 风险标记：设备端内存分配初始化，仅 2 GPU 测试覆盖，多进程测试偶发失败，影响所有量化级别

关联脉络

- 暂无明显关联 PR