PR #13397 完整报告

sgl-project/sglang

[sgl-kernel][CPU] add kernel for shm_allgather_into_tensor and shm_reduce_scatter_tensor

合并时间：2026-07-23 09:19

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/13397

执行摘要

- 一句话：新增 CPU 共享内存 allgather_into_tensor 和 reduce_scatter_tensor 内核
- 推荐动作：值得精读。重点了解：如何通过 STATE_GROUP 模板参数避免竞争条件；allreduce_workspace 的偏移量宏扩展机制；测试文件从 manual 迁至 registered 并注册 CI 的流程。

功能与动机

在 CPU 上实现 DP attention 需要高效的 allgather 和 reduce_scatter 原语，PR #12961 中的慢路径将被这两个内核替换。此外，现有 naive_all_gather 函数因重用状态组和缓冲区，在多 rank 跨阶段场景下可能导致挂起或错误结果，需一并修复。

实现拆解

1. 隔离状态组与缓冲区：将 allreduce_workspace 中的 states 数组从 2 扩展到 5，为每个分布式操作分配独立状态槽位；通过 STATE_GROUP_* 常量（定义在 shm.h）索引，并用模板参数 STATE_GROUP 将 all_gather 函数模板化，确保不同操作使用互不干扰的状态机和双缓冲区。
2. 新增 shm_allgather_into_tensor：在 interface.cpp 中实现 shm_allgather_into_tensor 函数，调用模板化的 all_gather<STATE_GROUP_ALL_GATHER_INTO_TENSOR>，并将结果写入用户提供的输出张量（而非内核内部分配）。对应在 torch_extension_cpu.cpp 中注册为 sgl_kernel.shm_allgather_into_tensor。
3. 新增 shm_reduce_scatter_tensor：在 interface.cpp 中实现 shm_reduce_scatter_tensor 函数，调用新编写的 reduce_scatter_outer_loop（基于共享内存的 reduce-scatter 逻辑），仅支持 ReduceOp.SUM。在 shm.h 中声明该函数，并在 torch_extension_cpu.cpp 中注册。
4. 扩展共享内存数据结构：在 shm.cpp 中为三个新操作添加双缓冲区指针 (allgather_buffer, allgather_into_tensor_buffer, reduce_scatter_buffer)；增加 BUFFER2_OFFSET/BUFFER3_OFFSET/BUFFER4_OFFSET 宏计算偏移量，确保各操作缓冲区不重叠。
5. 测试与 CI 注册：将 test/manual/cpu/test_comm.py 迁移至 test/registered/cpu/test_comm.py，新增 test_all_gather_into_tensor 和 test_reduce_scatter_tensor 测试方法（对比 dist 标准实现输出），并通过

register_cpu_ci 注册到 CPU 基础 CI 套件。

关键文件：

- sgl-kernel/csrc/cpu/shm.cpp（模块 内核实现；类别 source；类型 core-logic；符号 coll_state, allreduce_workspace, BUFFER2_OFFSET, BUFFER3_OFFSET）：核心实现文件，修改了状态枚举、workspace 结构、缓冲区定义和初始化，是避免竞态的关键。
- test/registered/cpu/test_comm.py（模块 测试；类别 test；类型 rename-or-move；符号 all_gather_into_tensor_fn, reduce_scatter_tensor_fn, test_all_gather_into_tensor, test_reduce_scatter_tensor）：测试文件，新增两个测试函数并注册到 CI，验证内核正确性。
- sgl-kernel/csrc/cpu/interface.cpp（模块 接口层；类别 source；类型 core-logic；符号 shm_allgather_into_tensor, shm_reduce_scatter_tensor）：实现层桥接，新增两个接口函数 shm_allgather_into_tensor 和 shm_reduce_scatter_tensor。
- sgl-kernel/csrc/cpu/torch_extension_cpu.cpp（模块 接口层；类别 source；类型 core-logic）：注册新操作到 Torch 扩展，使 Python 端可调用。
- sgl-kernel/csrc/cpu/shm.h（模块 内核实现；类别 source；类型 core-logic；符号 STATE_GROUP_SYMMETRIC_ALLREDUCE, STATE_GROUP_DISTRIBUTED_ALLREDUCE, STATE_GROUP_ALL_GATHER, STATE_GROUP_ALL_GATHER_INTO_TENSOR）：头文件，定义 STATE_GROUP 常量和模板声明。

关键符号：shm_allgather_into_tensor, shm_reduce_scatter_tensor, all_gather (template with STATE_GROUP), reduce_scatter_outer_loop, shm_initialize (modified states initialization)

关键源码片段

sgl-kernel/csrc/cpu/shm.cpp

核心实现文件，修改了状态枚举、workspace结构、缓冲区定义和初始化，是避免竞态的关键。

```
// 枚举增加 reduce_scatter 状态，与 all_gather 原有状态共同构成完整状态集
enum coll_state {
  coll_begin = 0,
  coll_allreduce_naive__copy_in_done,
  coll_allreduce_naive__reduce_done,
  coll_alt1_allreduce_naive__copy_in_done,
  coll_alt2_allreduce_naive__copy_in_done,
  coll_alt1_allreduce_naive__reduce_done,
  coll_allgather_naive__copy_in_done,
  coll_alt1_allgather_naive__copy_in_done,
  coll_alt2_allgather_naive__copy_in_done,
  // 新增的 reduce_scatter 阶段状态
  coll_reduce_scatter_naive__copy_in_done,
  coll_reduce_scatter_naive__reduce_done,
  coll_alt1_reduce_scatter_naive__copy_in_done,
  coll_alt2_reduce_scatter_naive__copy_in_done,
};
```

```
// workspace 结构：states 从 2 扩展到 5，buffer 按需增加
struct allreduce_workspace {
  // 5 个状态组分别对应：symmetric_allreduce, distributed_allreduce, all_gather, all_gather_into_
  tensor, reduce_scatter
  enum coll_state states[5];
  // 缓冲区布局：先两个 reduce 区，再三个 gather/scatter 区，每区均为双缓冲
  char buffer[
    2 * NAIVE_ALLREDUCE_THRESHOLD + // 对称 allreduce（小消息）
    2 * MAX_BUF_SIZE + // 分布式 allreduce（大消息）
    2 * MAX_BUF_SIZE + // allgather
    2 * MAX_BUF_SIZE + // allgather_into_tensor
    2 * MAX_BUF_SIZE // reduce_scatter
  ];
};

// 初始化时使用带语义的常量索引，提高可读性和维护性
workspace_buf->states[STATE_GROUP_SYMMETRIC_ALLREDUCE] = coll_alt2_allreduce_naive__
copy_in_done;
workspace_buf->states[STATE_GROUP_DISTRIBUTED_ALLREDUCE] = coll_begin;
// 新增状态组初始化为 coll_begin
for (int g = STATE_GROUP_ALL_GATHER; g <= STATE_GROUP_REDUCE_SCATTER; ++g) {
  workspace_buf->states[g] = coll_begin;
}
```

test/registered/cpu/test_comm.py

测试文件，新增两个测试函数并注册到 CI，验证内核正确性。

```
# 对比标准 dist.all_gather_into_tensor 的结果
@register_cpu_ci(est_time=30, suite="base-b-test-cpu")
def all_gather_into_tensor_fn(rank, world_size):
    for dtype in [torch.float32, torch.bfloat16, torch.float16]:
        tensor = torch.randn(2, 10, dtype=dtype)
        output_size = (tensor.size(0) * world_size,) + tensor.size()[1:]
        output_tensor = torch.empty(output_size, dtype=dtype)
        output_shm = torch.empty(output_size, dtype=dtype)
        # 使用标准分布式库作为参考
        dist.all_gather_into_tensor(output_tensor, tensor)
        # 调用 sgl 内核
        torch.ops.sgl_kernel.shm_allgather_into_tensor(output_shm, tensor)
        # 验证一致性
        torch.testing.assert_close(output_tensor, output_shm)

# 对比标准 dist.reduce_scatter_tensor 的结果
@register_cpu_ci(est_time=30, suite="base-b-test-cpu")
def reduce_scatter_tensor_fn(rank, world_size):
    op = dist.ReduceOp.SUM
    for dtype in [torch.float32, torch.bfloat16, torch.float16]:
        N, D = 4, 10
        tensor = torch.randn(world_size * N, D, dtype=dtype)
```

```
        output_tensor = torch.empty(N, D, dtype=dtype)
        output_shm = torch.empty_like(output_tensor)
        dist.reduce_scatter_tensor(output_tensor, tensor, op=op)
        torch.ops.sgl_kernel.shm_reduce_scatter_tensor(output_shm, tensor, op)
        torch.testing.assert_close(output_tensor, output_shm)

class TestComm(CustomTestCase):
    def test_all_gather_into_tensor(self):
        self._spawn_and_check(all_gather_into_tensor_fn)

    def test_reduce_scatter_tensor(self):
        self._spawn_and_check(reduce_scatter_tensor_fn)
```

评论区精华

本 PR 无 review 评论，由 mingfeima 直接批准。关键设计决策已在 PR body 中阐明：使用模板参数 STATE_GROUP 区分不同操作的状态索引，避免状态污染；扩展 allreduce_workspace 结构体以容纳 5 个状态组和对应双缓冲区；shm_reduce_scatter_tensor 仅支持 SUM 归约以保持简单。

- 暂无高价值评论线程

风险与影响

- 风险：新增内核依赖共享内存同步，若缓冲区分配或偏移量计算错误可能导致越界访问或死锁。wait_buffer_state_until_2 中的忙等待机制在极端竞争下可能有性能抖动（CPU 场景可接受）。shm_reduce_scatter_tensor 仅支持 SUM 归约，使用其他 op 会触发断言失败。测试覆盖了三种数据类型和 2 rank 场景，但未覆盖多 rank（>2）或异常路径。
- 影响：对用户：CPU 上的 DP attention 将获得专用通信内核，替代通用慢路径，预期提升性能和可扩展性。对系统：增加共享内存分配大小（每个操作额外 2 * MAX_BUF_SIZE），但通过双缓冲消除了操作间干扰。对团队：为后续添加更多 CPU collectives 提供了模板化模式，并建立了 CI 测试基线。
- 风险标记：竞态条件修复，共享内存竞争，仅支持 SUM 归约

关联脉络

- PR #12961 DP attention on CPU: 此 PR 提供的内核将替换 #12961 中的慢路径，是 DP attention 的核心依赖。