PR #46703 完整报告

vllm-project/vllm

[PERF] Extend NCCL symmetric memory to AllGather and ReduceScatter

合并时间：2026-07-01 02:38

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/46703

执行摘要

- 一句话：NCCL 对称内存扩展至 AllGather 与 ReduceScatter
- 推荐动作：值得精读：展现了将硬件加速（NCCL symmetric memory/NVLS）从单一操作扩展到多个 collectives 的工程模式。关注 _get_symm_scratch 的持久化设计、per-communicator 隔离的修复方式，以及 review 中关于 uniform size 标准化的设计讨论。如有 EP 场景，建议跟进修复 reduce_scatterv 的标准化问题。

功能与动机

Enable NCCL symmetric memory and NVLS (NVLink SHARP) for AllGather and ReduceScatter collectives (used by MoE dispatch/combine in Expert Parallel deployments), extending the existing symmetric-memory support that was limited to AllReduce. Also fixes per-communicator window registration so multiple NCCL communicators can share the symmetric memory pool without double-registering segments.

实现拆解

1. 修改 CudaCommunicator 路由逻辑：重写 all_gather、reduce_scatter、reduce_scatterv 方法，当启用对称内存且操作为 dim=0 uniform 时，转发到新增的 _all_gather_symm_mem 和 _reduce_scatter_symm_mem 私有方法；否则回退到 PyNCCL 或基类实现。新增 _get_symm_scratch 管理持久化的对称内存 scratch buffer，每个 rank 分配并注册一次。
2. 新增决策函数：在 all_reduce_utils.py 中添加 should_nccl_symm_mem_ag_rs，用于快速判断是否启用 AG/RS 对称内存路径（仅检查环境变量和 batch invariant），与已有的 should_nccl_symm_mem_allreduce 的 tensor-size 权衡逻辑不同。
3. 修复多 communicator 窗口注册：在 pynccl_allocator.py 中将全局集合 _registered_base_addrs 从 set[address] 改为 dict[bytes, set]，以 NCCL comm unique id 隔离不同通信器的已注册地址，防止重复注册导致 NVLS 失败。
4. 整合并扩展测试：删除单文件 test_nccl_symm_mem_allreduce.py，新建 test_nccl_symm_mem.py 包含 allreduce/allgather/reduce_scatter 三类测试，每个测试在独立进程中 spawn 多 rank worker 验证正确性。
5. 更新 CI 配置：将 .buildkite/test_areas/distributed.yaml 中的测试命令指向新文件。

关键文件：

- vllm/distributed/device_communicators/cuda_communicator.py（模块 通信层；类别 source；类型 core-logic；符号 all_gather, _get_symm_scratch, _reduce_scatter_symm_mem, _all_gather_symm_mem）：核心修改文件，实现所有新方法并修改路由逻辑
- tests/distributed/test_nccl_symm_mem.py（模块 测试；类别 test；类型 test-coverage；符号 nccl_symm_mem_allreduce_worker, test_nccl_symm_mem_allreduce, nccl_symm_mem_allgather_worker, test_nccl_symm_mem_allgather）：新增整合测试文件，覆盖所有三种操作
- tests/distributed/test_nccl_symm_mem_allreduce.py（模块 测试；类别 test；类型 deletion；符号 nccl_symm_mem_allreduce_worker, test_nccl_symm_mem_allreduce）：删除旧测试文件，内容已合并到新文件
- vllm/distributed/device_communicators/all_reduce_utils.py（模块 通信层；类别 source；类型 core-logic；符号 should_nccl_symm_mem_ag_rs）：新增 should_nccl_symm_mem_ag_rs 决策函数
- vllm/distributed/device_communicators/pynccl_allocator.py（模块 通信层；类别 source；类型 core-logic）：修复多 comm 窗口注册重复问题
- .buildkite/test_areas/distributed.yaml（模块 CI 配置；类别 config；类型 configuration）：CI 配置更新，指向新测试文件

关键符号：all_gather, _get_symm_scratch, _reduce_scatter_symm_mem, _all_gather_symm_mem, _all_gather_batched_symm_mem, should_nccl_symm_mem_ag_rs, nccl_symm_mem_allreduce_worker, nccl_symm_mem_allgather_worker, nccl_symm_mem_reduce_scatter_worker, test_nccl_symm_mem_allreduce, test_nccl_symm_mem_allgather, test_nccl_symm_mem_reduce_scatter

关键源码片段

vllm/distributed/device_communicators/cuda_communicator.py

核心修改文件，实现所有新方法并修改路由逻辑

```
# vllm/distributed/device_communicators/cuda_communicator.py

class CudaCommunicator(DeviceCommunicatorBase):
    # ... (init, all_reduce unchanged) ...

    def all_gather(self, input_: torch.Tensor, dim: int = -1) -> torch.Tensor:
        # Route uniform dim-0 all-gathers through NVLS symmetric memory when
        # enabled (mirrors reduce_scatter); otherwise fall back to the
        # base-class ring all-gather. Sequence parallelism's gather-before-GEMM
        # uses dim=0 with tp-aligned (uniform) shards.
        if dim < 0:
            dim += input_.dim()
        if dim == 0 and should_nccl_symm_mem_ag_rs():
```

```
            # NVLS requires contiguous tensors
            return self._all_gather_symm_mem(input_.contiguous())
        return super().all_gather(input_, dim)

    def reduce_scatter(self, input_: torch.Tensor, dim: int = -1):
        world_size = self.world_size
        pynccl_comm = self.pynccl_comm
        assert pynccl_comm is not None
        if dim < 0:
            dim += input_.dim()
        # movedim to make dim-0 the scatter dimension
        input_tensor = input_.movedim(0, dim).contiguous()
        assert input_tensor.shape[0] % world_size == 0
        chunk_size = input_tensor.shape[0] // world_size
        output_shape = (chunk_size,) + input_tensor.shape[1:]

        if should_nccl_symm_mem_ag_rs():
            output = self._reduce_scatter_symm_mem(input_tensor)
        else:
            output = torch.empty(output_shape, dtype=input_tensor.dtype,
                                 device=input_tensor.device)
            pynccl_comm.reduce_scatter(output, input_tensor)

        return output.movedim(0, dim).contiguous()

    def reduce_scatterv(self, input_: torch.Tensor, dim: int = -1,
                        sizes: list[int] | None = None):
        # ... dimension normalization ...
        # Symmetric memory is only used when all ranks have uniform sizes.
        # ncclCommWindowRegister is collective: asymmetric pool allocations
        # from variable per-rank sizes cause deadlocks.
        use_symm_mem = sizes is None and should_nccl_symm_mem_ag_rs()
        if use_symm_mem:
            output = self._reduce_scatter_symm_mem(input_tensor)
        else:
            output = torch.empty(output_shape, dtype=input_tensor.dtype,
                                 device=input_tensor.device)
            if sizes is not None and sizes.count(sizes[0]) != len(sizes):
                pynccl_comm.reduce_scatterv(output, input_tensor, sizes=sizes)
            else:
                pynccl_comm.reduce_scatter(output, input_tensor)
        return output.movedim(0, dim).contiguous()
```

tests/distributed/test_nccl_symm_mem.py

新增整合测试文件，覆盖所有三种操作

```
# tests/distributed/test_nccl_symm_mem.py

def nccl_symm_mem_allgather_worker(local_rank: int, world_size: int):
```

```
    monkeypatch = pytest.MonkeyPatch()
    with monkeypatch.context() as m:
        m.delenv("CUDA_VISIBLE_DEVICES", raising=False)
        dtype = torch.bfloat16
        device = torch.device(f"cuda:{local_rank}")
        torch.accelerator.set_device_index(device)
        torch.set_default_device(device)
        torch.set_default_dtype(dtype)
        update_environment_variables({
            "RANK": str(local_rank),
            "LOCAL_RANK": str(local_rank),
            "WORLD_SIZE": str(world_size),
            "MASTER_ADDR": "localhost",
            "MASTER_PORT": "12346",
        })

        init_distributed_environment()
        with ensure_current_vllm_config():
            initialize_model_parallel(tensor_model_parallel_size=world_size)

        # 获取 CudaCommunicator 实例
        cuda_communicator = typing.cast(
            CudaCommunicator, get_tp_group().device_communicator)
        if get_nccl_mem_pool() is None:
            pytest.skip("NCCL allocator compilation failed")
        if not is_symmetric_memory_enabled():
            pytest.skip("NCCL symmetric memory is disabled.")

        # 每个 rank 准备本地输入，然后通过 CudaCommunicator 的 all_gather 进行集合通信
        per_rank_size = test_size_elements
        input_tensor = torch.rand(per_rank_size, dtype=dtype, device=device)
        gathered = cuda_communicator.all_gather(input_tensor, dim=0)

        # 验证：使用 PyTorch 原生 all_gather 作为参考
        ref_list = [torch.empty_like(input_tensor) for _ in range(world_size)]
        dist.all_gather(ref_list, input_tensor, group=get_tp_group().device_group)
        ref = torch.cat(ref_list, dim=0)
        torch.testing.assert_close(gathered, ref, atol=2.5, rtol=0.1)


@pytest.mark.skipif(not current_platform.is_cuda(), reason="CUDA only")
@pytest.mark.parametrize("world_size", [2])
@pytest.mark.skipif(envs.VLLM_TARGET_DEVICE not in ["cuda"], reason="CUDA only")
def test_nccl_symm_mem_allgather(monkeypatch, world_size):
    if world_size > torch.accelerator.device_count():
        pytest.skip("Not enough GPUs")
    # 启用 SymmMemCommunicator 和相关环境变量
    monkeypatch.setenv("VLLM_USE_NCCL_SYMM_MEM", "1")
    monkeypatch.setenv("NCCL_NVLS_ENABLE", "1")
```

```
    monkeypatch.setenv("NCCL_CUMEM_ENABLE", "1")
    mp.spawn(nccl_symm_mem_allgather_worker, args=(world_size,), nprocs=world_size)
    cleanup_dist_env_and_memory()
```

vllm/distributed/device_communicators/all_reduce_utils.py

新增 should_nccl_symm_mem_ag_rs 决策函数

```
# vllm/distributed/device_communicators/all_reduce_utils.py

def should_nccl_symm_mem_ag_rs() -> bool:
    """Check whether NCCL symmetric memory should be used for
    AllGather / ReduceScatter collectives."""
    from vllm.distributed.device_communicators.pynccl_allocator import (
        is_symmetric_memory_enabled,
    )
    # batch invariant mode disables custom collectives
    if envs.VLLM_BATCH_INVARIANT:
        return False
    # 仅依赖全局是否启用对称内存，不进行 tensor size 权衡
    return is_symmetric_memory_enabled()
```

评论区精华

claude[bot] 在 reduce_scatterv 的 diff 上指出：该函数在 sizes 非 None 时直接跳过 NVLS 路径，而 all_gatherv 会将 uniform sizes 标准化为 None 从而走 NVLS 路径。在 EP combine 流中（all2all.py:137），sizes 总是提供且 steady-state 下 uniform，导致 combine 的 reduce_scatter 无法利用 NVLS，抵消了本次 PR 的部分收益。建议在 reduce_scatterv 顶部添加相同的 `if sizes is not None and all(s == sizes[0] for s in sizes): sizes = None` 标准化逻辑。该评论未在合并前得到明确回应，当前代码仍未包含该标准化，可能需后续修复。

- reduce_scatterv 中 uniform sizes 未标准化导致 NVLS 路径绕过 (design): PR 已合并但代码中未包含该标准化，可能需后续 PR 修复。

风险与影响

- 风险：
  - 回归风险：修改了核心通信路径 all_gather、reduce_scatter、reduce_scatterv，条件判断错误可能导致结果错误或死锁。
  - 兼容性：所有变化在 VLLM_USE_NCCL_SYMM_MEM=1 后，默认关闭，不影响现有用户。
  - 窗口注册修复：将全局集合改为 per-comm 字典，可能影响之前多 communicator 场景下的 memory pool 共享行为，但更正确。
  - 未解决问题：reduce_scatterv 的 uniform size 标准化缺失可能在实际 EP 场景中绕过 NVLS，降低部分收益。
- 影响：

- 用户：默认无影响；启用后，使用 Expert Parallelism 且 NVLS 兼容硬件的用户可显著提升 MoE dispatch/combine 性能。
- 系统：依赖 NVLink SHARP（NCCL_NVLS_ENABLE=1）和 CUDMEM（NCCL_CUMEM_ENABLE=1）支持。
- 团队：需要维护两个决策函数（AR vs AG/RS），并确保后续优化不破坏优先级顺序。
- 风险标记：核心通信路径变更，默认关闭降低风险，讨论中提及未解决问题

关联脉络

- PR #38549 [Original] NCCL symmetric memory AllReduce (squashed base): 本 PR 是其 squashed rebase，将对称内存从 AllReduce 扩展到 AllGather/ReduceScatter，并包含了其修复。