PR #24510 完整报告

sgl-project/sglang

Support multi-rank exchange via all_gather_object in dumper grafter

合并时间：2026-05-06 16:57

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24510

执行摘要

- 一句话：支持 dumper grafter 多 rank 全收集交换
- 推荐动作：值得精读，尤其是 maybe_intercept 的改进和多 rank 测试的实现方式。该 PR 展示了如何从简单 broadcast 迁移到 all_gather 以实现多 rank 收集，以及如何使用 CPU 多进程模拟多 rank 环境进行测试。

功能与动机

原有 1+1 broadcast 无法支持多个 sender rank 向多个 receiver rank 进行张量移植的场景。通过改为 all_gather_object，可以一次收集所有 sender 的贡献，并在 receiver 端切片处理。

实现拆解

1. 修改 _Grafter.maybe_intercept 方法（python/sglang/srt/debug_utils/dumper.py）：将 broadcast_object_list（1+1 模式）替换为 all_gather_object。发送方贡献自身 value，接收方贡献 None；接收方通过新增的 _sender_slice 方法从 gathered 列表中提取 sender 部分的张量列表，然后进行设备对齐、用户变换和 copy_。
2. 新增 _sender_slice 方法（dumper.py）：根据方向（B2T/T2B）从 gathered 列表中切分出属于 sender 的条目（B2T 取前 baseline_world_size 个元素，T2B 取后 target_world_size 个元素）。
3. 更新 _default_transform 方法（dumper.py）：使其能够处理多 sender 列表，现在尝试通过接收方自己在世界中的 rank 从 sender 列表中选择对应位置的张量（要求 sender 数量等于 receiver 世界大小）。若不满足则报错并提示用户提供自定义 transform。
4. 新增测试辅助函数（test/registered/debug_utils/test_dumper.py）：_run_graft_test_cpu_multi（使用 torch.multiprocessing 生成多个 CPU 进程）、_graft_cpu_worker_entry（每个 worker 初始化 gloo 进程组并执行工作函数）、_make_multi_rank_config（构建 DumperConfig 用于多 rank 场景）。
5. 新建测试类 TestGrafterMultiRankCpu（test_dumper.py）：包含两个测试用例：test_4_baseline_2_target_b2t_with_user_transform（4 baseline 发送、2 target 接收，b2t 方向）和 test_2_target_4_baseline_t2b_with_user_transform（2 target 发送、4 baseline 接收，t2b 方向），覆盖不对称多 rank 场景。

关键文件：

- python/sglang/srt/debug_utils/dumper.py（模块 调试工具；类别 source；类型 core-logic；符号 _sender_slice, maybe_intercept, _default_transform）：核心逻辑变更：通信原语从 broadcast 改为 all_gather，新增 _sender_slice 方法，更新默认转换以支持多 rank。
- test/registered/debug_utils/test_dumper.py（模块 测试；类别 test；类型 test-coverage；符号 _run_graft_test_cpu_multi, _graft_cpu_worker_entry, _make_multi_rank_config, TestGrafterMultiRankCpu）：添加了多 rank 测试覆盖，包括异步多进程启动和两个不对称 rank 测试用例。

关键符号：_Grafter.maybe_intercept, _Grafter._sender_slice, _Grafter._default_transform, _run_graft_test_cpu_multi, _graft_cpu_worker_entry, _make_multi_rank_config, TestGrafterMultiRankCpu.test_4_baseline_2_target_b2t_with_user_transform, TestGrafterMultiRankCpu.test_2_target_4_baseline_t2b_with_user_transform

关键源码片段

python/sglang/srt/debug_utils/dumper.py

核心逻辑变更：通信原语从 broadcast 改为 all_gather，新增 _sender_slice 方法，更新默认转换以支持多 rank。

```
def maybe_intercept(self, *, value, tags: dict) -> None:
    # ... 前面的逻辑 ...
    # all-gather over the graft world; sender ranks contribute `value`,
    # recv ranks contribute None (their local target is private and
    # shouldn't leak). all_gather_object is pickle-routed, so tensor
    # shapes may differ across sender ranks.
    total_world = cfg.grafter_baseline_world_size + cfg.grafter_target_world_size
    my_contribution = value if is_send else None
    gathered: list = [None] * total_world
    dist.all_gather_object(gathered, my_contribution, group=self._pg)

    if is_send:
        _log(f"[Grafter] send role={role.value} dir={direction.value} tags={tags}")
        return

    sender_contribs = self._sender_slice(direction=direction, gathered=gathered)
    # 将 pickled CUDA 张量恢复到本地设备
    sender_tensors = [
        (t.to(value.device) if isinstance(t, torch.Tensor) else t)
        for t in sender_contribs
    ]
    try:
        value_to_override = self._apply_transform(sender_tensors, target=value)
        _log(
            f"[Grafter] recv role={role.value} dir={direction.value} "
            f"tags={tags} n_senders={len(sender_tensors)}"
        )
```

```
            value.copy_(value_to_override)
        except Exception as e:
            _log(
                f"[Grafter] recv role={role.value} dir={direction.value} "
                f"tags={tags} transform/copy_ raised {type(e).__name__}: {e}; "
                f"skipping graft for this call (target tensor unchanged)\n"
                f"{traceback.format_exc()}"
            )

    def _sender_slice(self, *, direction: "_GraftDirection", gathered: list) -> list:
        """根据方向从 gathered 列表中切分出 sender 贡献的部分。"""
        cfg = self._config
        if direction == _GraftDirection.B2T:
            return gathered[: cfg.grafter_baseline_world_size]
        return gathered[cfg.grafter_baseline_world_size :]
```

test/registered/debug_utils/test_dumper.py

添加了多 rank 测试覆盖，包括异步多进程启动和两个不对称 rank 测试用例。

```
    def _run_graft_test_cpu_multi(
        worker_func, *, baseline_world: int, target_world: int, **kwargs
    ):
        """使用 CPU/gloo 后端启动多进程，模拟多 rank 场景。"""
        import torch.multiprocessing as mp

        role_ports = {
            "baseline": find_available_port(29800),
            "target": find_available_port(29900),
        }

        ctx = mp.get_context("spawn")
        result_queue = ctx.Queue()
        processes = []
        total = baseline_world + target_world
        for global_rank in range(total):
            if global_rank < baseline_world:
                role = "baseline"
                local_rank = global_rank
                local_world = baseline_world
            else:
                role = "target"
                local_rank = global_rank - baseline_world
                local_world = target_world
            p = ctx.Process(
                target=_graft_cpu_worker_entry,
                args=(role, local_rank, local_world, role_ports[role], worker_func, result_queue,
                kwargs),
            )
            p.start()
```

```
        processes.append(p)

    for p in processes:
        p.join()

    errors = [result_queue.get() for _ in range(total)]
    errors = [e for e in errors if e]
    if errors:
        raise AssertionError("\n".join(errors))


class TestGrafterMultiRankCpu:
    """覆盖不对称多 rank 情况（CI 只有 2 个 GPU，不足以测试这些场景）。"""

    def test_4_baseline_2_target_b2t_with_user_transform(self, tmp_path: Path):
        """4 个 baseline 发送 -> 2 个 target 接收，使用 b2t 方向。"""
        module_name = "_xform_assert_4_senders"
        (tmp_path / f"{module_name}.py").write_text(
            "import torch\n"
            "def transform(received_list, target):\n"
            "    # 自定义转换：将 sender 内容平均合并到 target 形状\n"
            "    return received_list[0]  # 此处简化\n"
        )
        # 启动多进程测试，验证 graft 正确性
```

评论区精华

审查者 gemini-code-assist[bot] 在 dumper.py 的行 172 建议将配置验证中的 assert 替换为显式的 if 检查和 ValueError，因为 assert 语句在 Python 优化模式（-O）下会被跳过，可能导致绕过验证。该建议尚未被采纳或讨论。

- 配置验证应使用 if 检查而非 assert (correctness): 尚未回复或解决。

风险与影响

- 风险：风险较低。主要风险是通信原语变更可能影响已有的一对一 graft 场景，但现有测试（如 test_copy_failure）仍使用 broadcast 原语，并未改为 all_gather，可能存在不一致。另外，新的多 rank 测试仅覆盖 CPU/gloo 后端，GPU/NCCL 后端下的多 rank 行为尚未测试。
- 影响：直接影响 dumper grafter 功能，允许用户在多 rank 环境中进行张量移植。对现有单 rank 用户透明（测试仍在）。对系统无性能影响（仅在调试时启用）。
- 风险标记：GPU 多 rank 未测试，assert 验证可能被跳过

关联脉络

- PR #24511 Support per-call extras and dataclass transform input in dumper grafter: 同一功能线（dumper grafter）的早期 PR，为本次多 rank 交换提供了基础。

- PR #24512 Enhance diff and tensor-info logging in dumper grafter: 同一功能线的增强日志 PR。
- PR #24513 Add e2e test with log snapshot in dumper grafter: 同一功能线添加端到端测试的 PR。