PR #21944 完整报告

sgl-project/sglang

Fix:fix(timeout): fix timeout not propagated

合并时间：2026-04-23 03:48

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/21944

执行摘要

- 一句话：修复分布式超时参数未传递到模型并行子组的问题
- 推荐动作：该 PR 是重要的 bugfix，涉及分布式核心路径，值得精读以理解超时传递机制。关注 _MODEL_PARALLEL_GROUP_TIMEOUT 全局变量的引入和传递方式，以及不同后端（mooncake、默认 NCCL）的统一处理。

功能与动机

根据关联 Issue #21911 的描述，当前 --dist-timeout 参数仅传递给 torch.distributed.init_process_group，但 SGLang 后续为模型并行（如 TP、ATTN_TP、MOE_DP 等）创建的 NCCL 子组未继承此超时设置。这导致即使设置了超时（如 3600 秒），子组集合操作仍可能因使用 PyTorch NCCL 后端默认的 10 分钟超时而失败，引发类似 "Watchdog caught collective operation timeout"的错误。

实现拆解

1. 引入全局超时变量：在 python/sglang/srt/distributed/parallel_state.py 中新增模块级变量 _MODEL_PARALLEL_GROUP_TIMEOUT，用于存储用户提供的超时值，类型为 Optional[timedelta]，初始化为 None。
2. 在初始化时设置超时：在 init_distributed_environment 函数中，当初始化全局进程组时，将传入的 timeout 参数转换为 timedelta 后赋值给 _MODEL_PARALLEL_GROUP_TIMEOUT，确保后续子组创建可复用。
3. 在子组创建时传递超时：在 _ModelParallelGroup 类的 __init__ 方法中，为每个子组（包括 mooncake、mooncake-cpu 及默认后端）的 torch.distributed.new_group 调用添加 timeout=subgroup_timeout 参数，其中 subgroup_timeout 从全局变量获取。
4. 清理时重置变量：在 destroy_distributed_environment 函数中，将 _MODEL_PARALLEL_GROUP_TIMEOUT 重置为 None，避免残留状态影响后续运行。

关键文件：

- python/sglang/srt/distributed/parallel_state.py（模块 分布式；类别 source；类型 core-logic；符号 _MODEL_PARALLEL_GROUP_TIMEOUT, init_distributed_environment, _ModelParallelGroup.init, destroy_distributed_environment）：这是唯一变更的文件，包含分布式并行状态管理的核心逻辑，负责进程组初始化和子组创建。

关键符号：init_distributed_environment, _ModelParallelGroup.init, destroy_distributed_environment

关键源码片段

python/sglang/srt/distributed/parallel_state.py

这是唯一变更的文件，包含分布式并行状态管理的核心逻辑，负责进程组初始化和子组创建。

```
# 在模块顶部定义全局超时变量，用于存储用户提供的超时值
_MODEL_PARALLEL_GROUP_TIMEOUT: Optional[timedelta] = None

def init_distributed_environment(
    world_size: int,
    rank: int,
    local_rank: int,
    distributed_init_method: Optional[str],
    backend: str,
    timeout: Optional[int] = None,
    # ... 其他参数
):
    # ... 其他初始化代码
    if not torch.distributed.is_initialized():
        global _MODEL_PARALLEL_GROUP_TIMEOUT # 声明全局变量以修改
        # ... 参数验证
        if timeout is not None:
            assert isinstance(timeout, (int)), "timeout must be a number"
            assert timeout > 0, "timeout must be positive"
            timeout = timedelta(seconds=timeout) # 转换为 timedelta
            _MODEL_PARALLEL_GROUP_TIMEOUT = timeout # 存储到全局变量
        # ... 初始化全局进程组

class _ModelParallelGroup:
    def __init__(
        self,
        group_ranks: List[List[int]],
        torch_distributed_backend: str,
        gloo_timeout: timedelta,
        # ... 其他参数
    ):
        # ... 设备初始化代码
        for ranks in group_ranks:
            active_ranks = torch.ones(len(ranks), dtype=torch.int32, device=self.device)
            active_ranks_cpu = torch.ones(len(ranks), dtype=torch.int32)
            subgroup_timeout = _MODEL_PARALLEL_GROUP_TIMEOUT # 从全局变量获取超时
            if "mooncake" in torch_distributed_backend:
                from mooncake.ep import MooncakeBackendOptions
                # 为 mooncake 后端设备组和 CPU 组都传递超时
                device_group = torch.distributed.new_group(
                    ranks,
```

```
                backend="mooncake",
                pg_options=MooncakeBackendOptions(active_ranks),
                timeout=subgroup_timeout, # 传递超时
            )
            cpu_group = torch.distributed.new_group(
                ranks,
                backend="mooncake-cpu",
                pg_options=MooncakeBackendOptions(active_ranks_cpu),
                timeout=subgroup_timeout, # 传递超时
            )
        else:
            pg_options = get_torch_distributed_pg_options(group_name)
            # 为默认后端设备组传递超时
            device_group = torch.distributed.new_group(
                ranks,
                backend=torch_distributed_backend,
                pg_options=pg_options,
                timeout=subgroup_timeout, # 传递超时
            )
            # CPU 组使用 gloo 后端，已有 gloo_timeout 参数
            cpu_group = torch.distributed.new_group(
                ranks, backend="gloo", timeout=gloo_timeout
            )
        # ... 组属性设置

def destroy_distributed_environment():
    global _WORLD, _MODEL_PARALLEL_GROUP_TIMEOUT # 清理时重置全局变量
    if _WORLD:
        _WORLD.destroy()
        _WORLD = None
    _MODEL_PARALLEL_GROUP_TIMEOUT = None # 重置超时变量
    if torch.distributed.is_initialized():
        torch.distributed.destroy_process_group()
```

评论区精华

reviewer gemini-code-assist[bot] 指出，在 mooncake 后端中，device_group 已添加超时参数，但 mooncake-cpu 组尚未应用。作者在后续提交中已修复此问题，确保所有子组（包括 mooncake-cpu）都接收 subgroup_timeout。reviewer ch-wan 简单批准（LGTM），表明变更被接受。

- 超时参数传递到 mooncake-cpu 子组 (correctness): 作者在后续提交中修复了此问题，确保 mooncake-cpu 组也接收 subgroup_timeout。

风险与影响

- 风险：回归风险低：变更仅添加超时参数传递，未修改核心逻辑；但需确保超时值正确传递到所有后端（包括 mooncake、gloo 及默认 NCCL）。 兼容性风险：

torch.distributed.new_group 的 timeout 参数支持 timedelta 类型，与现有代码中 gloo_timeout（可能为 timedelta）类型一致，但需确认所有 PyTorch 版本均支持此参数。 性能影响：超时设置本身不影响性能，仅改变超时行为；但若超时设置过短，可能增加因超 时导致的失败频率。

- 影响：对用户：修复后，用户通过 --dist-timeout 设置的超时将正确应用于所有分布式子组，避免因默认超时导致的意外失败，提升大规模分布式训练的稳定性。 对系统：确保超时一致性，减少因超时配置不一致引发的调试复杂度。 对团队：此修复针对核心分布式模块，需在涉及模型并行的测试中验证超时行为。
- 风险标记：核心路径变更，缺少测试覆盖

关联脉络

- 暂无明显关联 PR