PR #20460 完整报告

sgl-project/sglang

[HiCache] Add synchronization for context parallelism

合并时间：2026-04-28 02:13

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/20460

执行摘要

- 一句话：为 HiCache 添加 Context Parallelism 同步支持
- 推荐动作：建议精读 cache_controller.py 中的同步组创建逻辑（去重、gloo 后端选择）和 hiradix_cache.py 中的 fallback 到 tp_group 的策略，这些设计可用于其他需要跨并行组同步的场景。

功能与动机

PR body 指出 HiCache previously synchronized state only within tp_group, which is no longer sufficient after the CP split. This could cause different CP ranks to make different decisions about prefetch completion/revoke, write-through ack handling, and host-cache updates. 作者在 issue 评论中确认若不修复，CP+HiCache 会在 30-40 分钟后崩溃。

实现拆解

1. 参数替换：在 HiCacheController.__init__ 中接收 attn_cp_group 和 attn_tp_group 进程组对象，替代原先的 attn_cp_rank 和 attn_cp_size 标量。
2. CP 同步 API：新增 get_attn_cp_rank_and_size 从进程组派生 rank/size；_create_prefetch_sync_groups 基于 attn_cp/tp 组去重创建 gloo 后端同步组；_all_reduce_prefetch_groups 在 storage prefetch 关键路径上执行 all-reduce。
3. HiRadixCache 同步：新增 _all_reduce_attn_groups 和 _barrier_attn_groups 方法，在 prefetch 完成 / 撤销、write-through ack 处理等决策点调用 CP-aware 同步，若无 CP 则回退到 tp_group。
4. Storage key 修正：在 mooncake_store.py 中移除 storage 后缀中的 attn_cp_rank，避免不同 CP rank 使用不同 cache key 导致数据分裂。
5. 依赖注入：修改 hybrid_pool_assembler.py、cache_init_params.py、scheduler.py 等文件，将 attn_cp_group/attn_tp_group 沿调用链传递到 HiCacheController 和 HiRadixCache。
6. 测试覆盖：新增 TestMooncakeBackendQwen330BCP2 测试类，验证 TP2+CP2+Mooncake 场景下的稳定性。

关键文件：

- python/sglang/srt/managers/cache_controller.py（模块 缓存控制器；类别 source；类型 entrypoint；符号 get_attn_cp_rank_and_size, _create_prefetch_sync_groups, _destroy_prefetch_sync_groups, _all_reduce_prefetch_groups）：核心入口，新增 CP 同步组创建和 all-reduce 方法，修改构造函数接收进程组参数。
- python/sglang/srt/mem_cache/hiradix_cache.py（模块 Radix 缓存；类别 source；类型 core-logic；符号 _all_reduce_attn_groups, _barrier_attn_groups）：HiRadixCache 新增 CP-aware all-reduce 和 barrier 方法，并在初始化时传递进程组。
- python/sglang/srt/mem_cache/hybrid_cache/hybrid_pool_assembler.py（模块 混合池组装；类别 source；类型 dependency-wiring）：引入 attn_cp_group/attn_tp_group 参数，沿调用链传递到 HybridCacheController，是依赖注入的关键节点。
- python/sglang/srt/mem_cache/storage/mooncake_store/mooncake_store.py（模块 Mooncake 存储；类别 source；类型 core-logic）：调整 suffix 生成逻辑，移除 attn_cp_rank 防止 cache key 分叉，消除 CP 对存储层的干扰。
- test/registered/hicache/test_hicache_storage_mooncake_backend.py（模块 测试用例；类别 test；类型 test-coverage；符号 TestMooncakeBackendQwen330BCP2, _get_model_name, _get_additional_server_args_and_env）：新增 TestMooncakeBackendQwen330BCP2 测试类，覆盖 TP2+CP2+Mooncake 组合场景。

关键符号：get_attn_cp_rank_and_size, _create_prefetch_sync_groups, _destroy_prefetch_sync_groups, _all_reduce_prefetch_groups, _all_reduce_attn_groups, _barrier_attn_groups

关键源码片段

python/sglang/srt/managers/cache_controller.py

核心入口，新增 CP 同步组创建和 all-reduce 方法，修改构造函数接收进程组参数。

```
# HiCacheController 类中的关键 CP 同步方法

def get_attn_cp_rank_and_size(self) -> tuple[int, int]:
    """从 attn_cp 进程组派生 CP rank 和 size。
    若未启用 CP 则返回 (0, 1) 作为单 rank 默认值。
    """
    if self.attn_cp_group is not None:
        return (
            torch.distributed.get_rank(group=self.attn_cp_group),
            torch.distributed.get_world_size(group=self.attn_cp_group),
        )
    return 0, 1

def _create_prefetch_sync_groups(self) -> None:
    """为 storage prefetch 创建 CP-aware 的同步组。
    使用 gloo 后端避免与 NCCL 流冲突。
    去重逻辑避免对同一 rank 集合重复创建。
    """
    from sglang.srt.distributed.parallel_state import create_custom_parallel_group
```

```
    self.prefetch_sync_groups = []
    seen_rank_sets = set()
    # 优先使用 attn_cp 和 attn_tp 组，若两者均未启用则回退到 tp_group
    if self.attn_cp_group is not None or self.attn_tp_group is not None:
        base_groups = [self.attn_cp_group, self.attn_tp_group]
    else:
        base_groups = [self.tp_group]
    for group in base_groups:
        if group is None or torch.distributed.get_world_size(group=group) == 1:
            continue # 单 rank 无需同步
        group_ranks = tuple(torch.distributed.get_process_group_ranks(group))
        if group_ranks in seen_rank_sets:
            continue # 跳过重复的 rank 组合
        seen_rank_sets.add(group_ranks)
        self.prefetch_sync_groups.append(
            create_custom_parallel_group(group_ranks=list(group_ranks), backend="gloo")
        )

def _destroy_prefetch_sync_groups(self) -> None:
    """清理 prefetch 同步组。"""
    for group in self.prefetch_sync_groups:
        try:
            torch.distributed.destroy_process_group(group)
        except Exception:
            pass
    self.prefetch_sync_groups = []

def _all_reduce_prefetch_groups(self, tensor: torch.Tensor, op) -> None:
    """在 prefetch 同步组上执行 all-reduce。"""
    for group in self.prefetch_sync_groups:
        torch.distributed.all_reduce(tensor, op=op, group=group)
```

评论区精华

- 性能影响：libratiger 询问是否提升性能，作者 vladnosiv 确认这是可靠性修复，没有性能提升。
- 测试要求：hzh0425 要求添加 CP+HiCache 组合测试，作者陆续添加了 qwen3-30b + CP + hicache + file/mooncake 和 dsv32 测试用例。
- CI 兼容：hzh0425 提到 HiCache 的 CI 因 CUDA 13 环境不兼容被临时移除，需要等待 CI 恢复。
- 性能影响 (question): 作者确认是可靠性修复，不提升性能。
- 测试覆盖 (testing): 作者添加了 qwen3-30b + CP + hicache + file/mooncake 等多个测试。
- CI 环境兼容 (infra): 等待 CI 恢复后继续。
- 冲突解决 (other): 修复了与 PR #21259 的冲突。

风险与影响

- 风险：
  1. 非 CP 场景回归：新增同步组创建逻辑通过 None 检查和单 rank 跳过，不影响非 CP 路径。
  2. 同步开销：all_reduce 和 barrier 仅在 storage prefetch 关键路径上调用，且使用 gloo 后端避免与 NCCL 流冲突。
  3. 数据兼容性：mooncake_store.py 中 suffix 修改可能影响已有 cache key 格式，需确保升级后旧数据可忽略或迁移。
  4. 测试覆盖不全：仅 Mooncake 后端有 CP 测试，File 后端未添加对应测试。 - 影响：对用户：只有同时启用 HiCache 和 Context Parallelism 的用户受影响，修复了稳定性问题。对系统：增加了少量同步开销，但确保了多 CP rank 间状态一致性。对团队：统一了进程组传递模式，未来可扩展支持其他并行维度。 - 风险标记：核心路径变更，新增同步开销，CP scenario only，缺少 File 后端 CP 测试

关联脉络

- PR #21259 [unknown]: 作者指出需要解决与此 PR 的冲突并重新测试。