PR #25395 完整报告

sgl-project/sglang

[UnifiedTree] Add CP sync

合并时间：2026-06-03 16:10

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25395

执行摘要

- 一句话：为 UnifiedRadixTree 添加 CP 同步并修复 DSv4 连续性
- 推荐动作：值得精读，特别是 _all_reduce_attn_groups 的设计权衡。建议关注该 PR 是否导致 2D 并行场景下的同步覆盖不足，后续可根据需要扩展为每组独立同步。

功能与动机

HiCache 需要与 Context Parallel (CP) 进行状态同步。此前 PR #20460 已为旧 RadixTree 添加该功能，现在需在 UnifiedRadixTree 中实现相同机制以确保缓存一致性。

实现拆解

1. 添加 CP/TP 分组属性：在 UnifiedRadixCache.__init__ 中从参数提取 attn_cp_group 和 attn_tp_group，分别对应 CP 和 TP 的通信组。
2. 新增通信辅助方法：
   - _all_reduce_attn_groups(tensor, op)：遍历 CP 和 TP 组执行 all_reduce，若均无有效组则回退到 tp_group。
   - _barrier_attn_groups()：遍历 CP 和 TP 组执行 barrier，同理回退。
3. 替换写入检查同步点：在 writing_check 中，将原来仅对 tp_group 的 all_reduce 替换为对 _all_reduce_attn_groups 的调用，使所有参与 2D 并行通信的组都能同步。
4. 修复 DSv4 张量连续性：在 deepseek_v4.py 的 _compute_kv_bf16 中，对 qkv_a 切片 .contiguous() 保证后续算子获取连续张量（CP 预填充路径）。
5. 新增集成测试：创建 test_unified_radix_cache_kl_cp.py，使用 Qwen3-30B-A3B-FP8 模型，以 4GPU + CP2 配置启动服务，通过 UnifiedRadixTreeTestMixin 运行 KL 散度、GSM8K、MMLU 等一致性测试，验证 CP+HiCache 组合的正确性。

关键文件：

- python/sglang/srt/mem_cache/unified_radix_cache.py（模块 缓存层；类别 source；类型 core-logic；符号 _all_reduce_attn_groups, _barrier_attn_groups）：核心修改文件：添加 CP/TP 同步方法并替换所有 TP 同步点。
- test/registered/radix_cache/unified_radix_tree/test_unified_radix_cache_kl_cp.py（模块 测试；类别 test；类型 test-coverage；符号 TestUnifiedQwen3HiCacheCP, setUpClass, tearDownClass）：新增集成测试文件，覆盖

HiCache+CP+UnifiedRadixTree 组合场景。

关键符号：_all_reduce_attn_groups, _barrier_attn_groups, TestUnifiedQwen3HiCacheCP, setUpClass, tearDownClass

关键源码片段

python/sglang/srt/mem_cache/unified_radix_cache.py

核心修改文件：添加 CP/TP 同步方法并替换所有 TP 同步点。

```
# python/sglang/srt/mem_cache/unified_radix_cache.py

class UnifiedRadixCache:
    def __init__(self, params):
        # ... 其他初始化 ...
        self.tp_group = params.tp_cache_group
        self.attn_cp_group = params.attn_cp_cache_group # 新增：CP 通信组
        self.attn_tp_group = params.attn_tp_cache_group # 新增：TP 通信组
        self.tp_world_size = (
            1
            if self.tp_group is None
            else torch.distributed.get_world_size(group=self.tp_group)
        )
        # ...

    def _all_reduce_attn_groups(self, tensor: torch.Tensor, op):
        """遍历 CP 和 TP 组执行 all_reduce，若均不可用则回退到 tp_group"""
        reduced = False
        for group in (self.attn_cp_group, self.attn_tp_group):
            if group is not None and torch.distributed.get_world_size(group=group) > 1:
                torch.distributed.all_reduce(tensor, op=op, group=group)
                reduced = True
        if not reduced and self.tp_world_size > 1:
            torch.distributed.all_reduce(tensor, op=op, group=self.tp_group)

    def _barrier_attn_groups(self):
        """遍历 CP 和 TP 组执行 barrier，若均不可用则回退到 tp_group"""
        waited = False
        for group in (self.attn_cp_group, self.attn_tp_group):
            if group is not None and torch.distributed.get_world_size(group=group) > 1:
                torch.distributed.barrier(group=group)
                waited = True
        if not waited and self.tp_world_size > 1:
            torch.distributed.barrier(group=self.tp_group)

    def writing_check(self, write_back: bool = False) -> None:
        # ... 原有逻辑 ...
        queue_size = torch.tensor(finish_count, dtype=torch.int, device="cpu")
        # 替换前：if self.tp_world_size > 1: all_reduce(queue_size, ...)
```

```
        self._all_reduce_attn_groups(queue_size, torch.distributed.ReduceOp.MIN)
        finish_count = int(queue_size.item())
        # ...
```

test/registered/radix_cache/unified_radix_tree/test_unified_radix_cache_kl_cp.py

新增集成测试文件，覆盖 HiCache+CP+UnifiedRadixTree 组合场景。

```
# test/registered/radix_cache/unified_radix_tree/test_unified_radix_cache_kl_cp.py

# 在 CI 注册该测试，估算时间 400s，基础阶段运行
register_cuda_ci(est_time=400, stage="base-c", runner_config="4-gpu-h100")

class TestUnifiedQwen3HiCacheCP(UnifiedRadixTreeTestMixin, CustomTestCase):
    """Qwen3-30B-FP8 + HiCache + CP + UnifiedRadixCache 集成测试"""
    # 配置参数
    hicache_io_backend = "direct"
    hicache_mem_layout = "page_first_direct"
    max_running_requests = 32
    kl_threshold = 0.005
    gsm8k_threshold = 0.7
    mmlu_threshold = 0.7

    @classmethod
    def setUpClass(cls):
        cls.model = QWEN3_30B_MODEL
        cls.base_url = DEFAULT_URL_FOR_TEST
        # 启动服务：4GPU、CP2、HiCache 开启、使用 UnifiedRadixTree
        cls.process = popen_launch_server(
            cls.model,
            cls.base_url,
            timeout=DEFAULT_TIMEOUT_FOR_SERVER_LAUNCH,
            other_args=[
                "--trust-remote-code",
                "--tp-size", "4",
                "--moe-dp-size", "1",
                "--ep-size", "4",
                "--attn-cp-size", "2", # CP size=2
                "--enable-prefill-context-parallel",
                "--mem-fraction-static", "0.8",
                "--cuda-graph-max-bs", "32",
                "--max-running-requests", str(cls.max_running_requests),
                "--disable-piecewise-cuda-graph",
                "--model-loader-extra-config",
                '{"enable_multithread_load": true, "num_threads": 64}',
                # HiCache 相关参数
                "--enable-hierarchical-cache",
                "--hicache-ratio", "2",
                "--hicache-write-policy", "write_through",
```

```
                "--hicache-io-backend", cls.hicache_io_backend,
                "--hicache-mem-layout", cls.hicache_mem_layout,
            ],
            env={"SGLANG_ENABLE_UNIFIED_RADIX_TREE": "1"}, # 启用 UnifiedRadixTree
        )
        cls.input_ids = get_input_ids(cls.model, num_samples=18)

    @classmethod
    def tearDownClass(cls):
        kill_process_tree(cls.process.pid)
```

评论区精华

- 同步逻辑互斥问题：gemini-code-assist[bot] 指出 _all_reduce_attn_groups 和 _barrier_attn_groups 中 CP 和 TP 的同步互斥——若 CP 组存在则跳过 TP 组，可能导致 2D 并行下 TP 维度未同步。建议分别对每个组独立执行 all_reduce/barrier。最终 PR 未采纳该建议，维持原有互斥逻辑。
- 测试跳过名称确认：gemini-code-assist[bot] 质疑跳过的测试名称与 PR 描述不符。PR body 提到 test_multiturn_decode_cache_hit_branching 失败，代码也跳过了该测试，但 bot 误以为应跳过 test_multiturn_prefill_cache_hit_branching。经确认跳转正确。
- DSv4 Bug 修复：vladnosiv 在 review 中说明 deepseek_v4.py 的 .contiguous() 修复是必要的，因为后续调用 _compute_kv_bf16 的方法已变为无操作（no-op），但该修复仍保证正确性。
  - 同步逻辑互斥问题 (correctness): PR 作者未修改，保留互斥逻辑，但合并者批准。可能存在风险。
  - 测试跳过名称确认 (question): 实际跳过正确（decode 版本），bot 误解。
  - DSv4 张量连续性修复 (bugfix): 已修复。

风险与影响

- 风险：
  - 同步互斥风险：如果环境同时启用 CP 和 TP 且 attn_tp_group 未分离，可能导致 TP 维度未同步，引发缓存状态不一致。当前逻辑在 CP 存在时跳过 TP all_reduce/barrier。
  - DSv4 回归：deepseek_v4.py 的修改虽小，但影响 CP 预填充分支，需确保非 CP 路径不受影响。
  - 测试稳定性：新测试使用 4 GPU 大模型，KL 阈值 0.005 较严格，CI 中曾因其他问题失败（rerun 后通过），存在不稳定性。
  - 配置依赖：测试依赖 attn-cp-size=2 和特定环境变量 SGLANG_ENABLE_UNIFIED_RADIX_TREE=1，配置变动可能导致测试失效。
- 影响：
  - 用户影响：使用 UnifiedRadixTree 且启用 HiCache+CP 的用户将获得正确的缓存同步；DSv4 用户可能间接受益于张量连续性修复。
  - 系统影响：新增同步通信会引入少量额外开销（两个 all_reduce/barrier 调用）。

- 团队影响：为 UnifiedRadixTree 的 CP 支持打下基础，后续维护者需注意 2D 并行同步逻辑的权衡。
- 风险标记：2D 并行互斥，测试稳定性，核心同步逻辑

关联脉络

- PR #24984 [HiCache] feat: support draft offload for mooncake: 都与 HiCache 集成相关，本 PR 为 UnifiedRadixTree 添加 CP 同步，以支持 HiCache 在 CP 场景下的正确性。