PR #27285 完整报告

sgl-project/sglang

[HiCache] Fix crash when using PP + HiCache L2

合并时间：2026-06-06 16:57

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27285

执行摘要

- 一句话：修复 PP+HiCache L2 崩溃并引入 pp_sync 同步机制
- 推荐动作：值得精读，尤其是 _pp_sync 在调度器层面实现 PP 同步的设计模式。关注 _reap_completed_async_work 的异步管理技巧，以及 writing_check 中基于 PP rank 的条件处理。对于使用 PP+HiCache 的团队，建议及时合并。

功能与动机

PR 描述指出 "HiCache does not work well with PP. It leads to server crash." 根本原因是 PP0 和 PP1 的调度线程间缺乏同步，具体细节在 #22607 中讨论。

实现拆解

1. 在 cache_init_params.py 中增加 pp_cache_group 字段，允许传递 PP 通信组到缓存层。
2. 在 hiradix_cache.py 和 unified_radix_cache.py 的 __init__ 中新增 pp_group、pp_rank、pp_size 属性及 work_list 列表，用于追踪异步发送操作。
3. 新增三个核心方法：_reap_completed_async_work() 清理已完成异步工作；_all_reduce() 在 PP0 上做 TP all-reduce 后通过 _pp_sync() 沿 PP 管道传播结果；_pp_sync() 使用 isend/recv 实现数据传递。
4. 修改 writing_check 逻辑，仅在 pp_rank == 0 时处理 ack_write_queue，然后通过 _all_reduce 同步计数，保证所有 PP rank 的缓存状态转换一致。
5. 调整 hybrid_pool_assembler.py、cache_controller.py、hybrid_cache_controller.py 的构造函数签名，传递 pp_group 替代 pp_rank/pp_size；在 cache_controller.py 中动态获取 PP rank/size。
6. 新增端到端测试文件 test_unified_radix_cache_hicache_pp_kl.py，使用 Qwen3-30B-A3B-FP8 模型在 tp=2/pp=2 配置下验证 GSM8K 准确率。

关键文件：

- python/sglang/srt/mem_cache/hiradix_cache.py（模块 HiCache 层；类别 source；类型 core-logic；符号 _reap_completed_async_work, _all_reduce, _pp_sync）：HiCache 主类，新增 pp_group、work_list 属性及 _pp_sync、_all_reduce、_reap_completed_async_work 方法，是同步机制核心实现。

- python/sglang/srt/mem_cache/unified_radix_cache.py（模块 统一缓存；类别 source；类型 core-logic；符号 _reap_completed_async_work, _all_reduce, _pp_sync）：UnifiedRadixCache 基类，同步机制与 hiradix_cache 对应，同时修改了 load_back_threshold 和 writing_check 逻辑。
- test/registered/radix_cache/unified_radix_tree/test_unified_radix_cache_hicache_pp_kl.py（模块 端到端测试；类别 test；类型 test-coverage；符号 _assert_pp_decode_cached_tokens, TestUnifiedQwen3HiCachePP, test_gsm8k, setUpClass）：新增端到端测试，验证 PP+HiCache 组合下 GSM8K 精度达标，确保修复有效性。
- python/sglang/srt/mem_cache/hybrid_cache/hybrid_pool_assembler.py（模块 缓存组装层；类别 source；类型 core-logic）：调整了 build_kv_only_stack、build_hybrid_swa_stack、build_deepseek_v4_hicache_stack 的签名，传递 pp_group 替代 pp_rank/pp_size。
- python/sglang/srt/managers/cache_controller.py（模块 缓存控制器；类别 source；类型 entrypoint）：更新构造函数签名和 storage 配置初始化，动态获取 PP rank/size，替代原先的静态参数。
- python/sglang/srt/mem_cache/hybrid_cache/hybrid_cache_controller.py（模块 混合缓存控制器；类别 source；类型 entrypoint）：签名变更，传递 pp_group 替代 pp_rank/pp_size。
- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 entrypoint）：调度器入口文件，新增对 _reap_completed_async_work 的调用（+1 行），确保每次调度循环前清理异步状态。
- python/sglang/srt/mem_cache/kv_cache_builder.py（模块 缓存构建器；类别 source；类型 core-logic）：传递 pp_group 到缓存构建函数。
- test/registered/unit/mem_cache/test_unified_radix_cache_unittest.py（模块 单元测试；类别 test；类型 test-coverage）：单元测试调整以适应缓存层新参数变化。
- python/sglang/srt/mem_cache/cache_init_params.py（模块 缓存参数；类别 source；类型 core-logic）：新增 pp_cache_group 字段，使得 PP 通信组可被传递到缓存层。

关键符号：_reap_completed_async_work, _all_reduce, _pp_sync, writing_check, _assert_pp_decode_cached_tokens, test_gsm8k, setUpClass, tearDownClass, _generate_storage_config

关键源码片段

python/sglang/srt/mem_cache/unified_radix_cache.py

UnifiedRadixCache 基类，同步机制与 hiradix_cache 对应，同时修改了 load_back_threshold 和 writing_check 逻辑。

```
    def _reap_completed_async_work(self):
        """
        轮询并收拢已完成异步工作，防止 work_list 无限增长。
        必须由调度线程调用。
        """
```

```
        count = 0
        while count < len(self.work_list) and self.work_list[count].is_completed():
            count += 1
        if count > 0:
            self.work_list = self.work_list[count:]

    # 在 writing_check 中，仅 PP0 消费 ack_write_queue
    if self.pp_rank == 0:
        for _, finish_event, ack_list in cc.ack_write_queue:
            if not finish_event.query():
                break
            finish_count += 1
    # 然后通过 all_reduce 确保所有 PP rank 持有相同的完成计数
    queue_size = torch.tensor(finish_count, dtype=torch.int, device="cpu")
    self._all_reduce(queue_size, torch.distributed.ReduceOp.MAX)
```

评论区精华

review 中主要讨论了命名和范围：

- hzh0425 建议将 pp_cache_group 改名为 attn_pp_cache_group 以保持与 attn_cp/attn_tp 的一致性，ShangmingCai 表示赞同，但作者 stepinto 在讨论后决定保持原样。
- hzh0425 指出 DeepSeekV4 相关变更应分离到单独 PR，stepinto 按要求回退了相关修改。
- pp_cache_group 命名讨论 (design): 作者 stepinto 回复保持当前命名，已达成一致。
- DSv4 变更分离 (design): stepinto 回退了 DSv4 相关修改，只保留通用 PP 同步。

风险与影响

- 风险：技术风险包括：
  - 死锁：_pp_sync 中 isend/recv 配对错误可能导致 PP 管道阻塞，work_list 未正确清理也可能造成内存泄漏或死锁。
  - L3 未覆盖：该 PR 仅修复 L2，用户若在 PP 下启用 L3 存储仍可能崩溃。
  - 性能影响：每次 writing_check 都执行 _all_reduce 会增加同步开销，高并发场景可能影响吞吐量。
  - 测试局限：仅测试了 Qwen3-30B-A3B-FP8 单一模型和 GSM8K 任务，未覆盖其他模型或长序列场景。
  - 影响：对用户：使用 PP + HiCache L2 的用户不再遇到崩溃，缓存命中率和 TTFT 显著改善（见 PR 中 benchmark 数据）。L3 用户暂不受益，需等待后续补丁。 对系统：新增的 PP 同步机制可能小幅增加调度延迟，但保障了正确性。 对团队：该 PR 拆分了原大型补丁，使剩下的 L3 支持工作更清晰，需继续跟进。
- 风险标记：PP 同步死锁风险，L3 未修复，测试覆盖有限，ALL reduce 性能开销

关联脉络

- PR #27010 [WIP] PP + HiCache L2+L3 combined fix: 该 PR 是 #27010 的拆分，只包含 L2 修复，L3 后续解决。
- PR #22607 Discussion on PP + HiCache synchronization: PR body 引用该 issue 作为同步问题细节的讨论背景。