PR #28916 完整报告

sgl-project/sglang

[HiCache] Fix hicache host memory leak by bounding PP-sync work_list

合并时间：2026-06-23 16:39

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/28916

执行摘要

- 一句话：修复 HiCache PP-sync 主机内存泄漏
- 推荐动作：PR 逻辑清晰，修复了明确的 bug，建议合入。代码变更量小，但 reviewer（bot）提出的 try...finally 建议值得考虑以增强健壮性。合入后建议部署到受影响的 PD 分离预填充服务验证内存增长曲线。

功能与动机

关联 Issue #28902 报告了 PP=0 服务器在启用 PP 和 HiCache 时主机内存以约 12GB/h 的速率泄漏，即使空闲时也持续增长。PR body 明确指出来源是 PP-sync 路径（#27285）中 work_list 无背压增长。

实现拆解

1. 替换非阻塞收割为阻塞排空：在 HiRadixCache 和 UnifiedRadixCache 中，将 _reap_completed_async_work 方法替换为 _drain_async_work。新方法遍历 work_list 中的每个 work 对象并调用其 wait() 阻塞等待，然后清空列表。
2. 调整调用时机：在 check_hicache_events() 中将 _drain_async_work 的调用从末尾移到最前面，确保在发起新一轮异步发送之前先排空上一轮的所有待处理发送，从而将 work_list 大小限制为单轮发送。
3. 添加单元测试：新增 test/registered/unit/mem_cache/test_hiradix_pp_sync_drain.py，使用伪造的 _FakeWork 和 _Holder 模拟 work_list，验证 _drain_async_work 会等待所有 work 完成并清空列表，以及空列表时也是无操作的。
4. 双模块同步修复：HiRadixCache（commit 5a2a0ee）和 UnifiedRadixCache（commit f818019）各自独立应用了相同的代码变更，确保两种缓存实现的一致性。

关键文件：

- python/sglang/srt/mem_cache/hiradix_cache.py（模块 HiCache；类别 source；类型 core-logic；符号 _reap_completed_async_work, _drain_async_work, check_hicache_events）：核心修复文件：替换 _reap_completed_async_work 为 _drain_async_work，并调整 check_hicache_events 中的调用顺序。
- python/sglang/srt/mem_cache/unified_radix_cache.py（模块 HiCache；类别 source；类型 core-logic；符号 _reap_completed_async_work, _drain_async_work, check_hicache_events）：相同修复应用于 UnifiedRadixCache，确保两种缓存实现的一致

性。

- test/registered/unit/mem_cache/test_hiradix_pp_sync_drain.py（模块 测试；类别 test；类型 test-coverage；符号 _FakeWork, init, wait, _Holder）：新增单元测试，覆盖正常排空和空列表场景，确保修复的正确性。

关键符号：_drain_async_work, check_hicache_events, _reap_completed_async_work

关键源码片段

python/sglang/srt/mem_cache/hiradix_cache.py

核心修复文件：替换 _reap_completed_async_work 为 _drain_async_work，并调整 check_hicache_events 中的调用顺序。

```
def _drain_async_work(self):
    """
    Block until all outstanding async sends are consumed, then clear.

    Called at the start of each event round, so work_list holds the sends
    accumulated since the last round. This bounds it and applies
    backpressure when a downstream PP rank lags. Scheduler thread only.
    """
    for work in self.work_list:
        work.wait() # 阻塞等待每个异步发送完成
    self.work_list.clear() # 清空列表，确保列表大小不会持续增长

def check_hicache_events(self):
    # 在发起新一轮发送之前，先排空上一轮的未完成发送
    self._drain_async_work()
    self.writing_check()
    self.loading_check()
    if self.enable_storage:
        self.drain_storage_control_queues()
    if self.enable_storage_metrics:
        self.storage_metrics_collector.log_storage_metrics(
            self.cache_controller.storage_backend.get_stats()
        )
```

评论区精华

Gemini Code Assist 提出了防御性编程建议：将 work_list 的遍历放在 try...finally 块中，以确保即使 wait() 抛出异常也能清空列表，避免缓存进入不一致状态。该建议未被作者采纳（保持简单实现），也未引起其他 reviewer 讨论。两个 reviewer（hzh0425、ShangmingCai）均批准了 PR。

- 防御性编程：使用 try...finally 确保 work_list 在异常时被清空 (correctness): 未被采用，当前实现简单，reviewer 未明确反对。

风险与影响

- 风险：
  - 回归风险：从非阻塞轮询改为阻塞等待，可能增加调度器主循环的延迟，尤其是当下游 PP 排名响应慢时。但这是有意为之的背压机制。
  - 死锁风险：如果 wait() 永远不返回（如分布式通信断裂），可能导致调度器线程阻塞。PR 未添加超时机制。
  - 兼容性：仅修改内部方法，对外接口和运行时配置无影响。
  - 测试覆盖：单元测试仅覆盖正常路径和空列表，缺少异常处理和超时场景。
- 影响：
  - 用户 / 运维：解决了 HiCache + PP 环境下主机内存泄漏问题，避免了服务因 OOM 被自动重启。
  - 系统：调度器事件循环中增加阻塞等待，可能轻微增加单轮事件处理时间，但换来了可预测的内存边界。
  - 团队：需要在释放说明中强调此修复依赖 HiCache 和 PP 同时启用的场景。
  - 影响程度：中（修复一个影响特定配置的严重内存泄漏）。
  - 风险标记：核心路径变更，缺少异常处理，缺少超时机制

关联脉络

- PR #27285 [HiCache] Add PP-sync path: 引入了 PP-sync 路径和 work_list 机制，当前 PR 修复了该机制导致的内存泄漏。
- PR #28902 [Bug] PP=0 server memory leaks when both PP and HiCache are enabled: 关联 Issue，报告了内存泄漏的具体现象，当前 PR 直接解决该 issue。