PR #7107 完整报告

PaddlePaddle/FastDeploy

[PD Disaggregation] Write the cache of preempted req to storage and refine PD Disaggregation

合并时间：2026-04-01 13:15

原文链接：http://prhub.com.cn/PaddlePaddle/FastDeploy/pull/7107

执行摘要

本 PR 优化了 FastDeploy 中 PD Disaggregation 场景下的抢占请求处理，通过将抢占请求的 KV cache 写入 storage 后端以提升缓存复用率，并调整调度逻辑避免死锁。关键变更包括新增环境变量控制、调度器写 cache 逻辑和缓存管理函数参数调整，影响系统性能和稳定性，但 review 中未解决锁内调用和类型一致性等风险，需后续关注。

功能与动机

根据 PR 作者 juncaipeng 的描述，此变更旨在解决三个核心问题：

- 优化抢占请求处理：当启用缓存池化时，抢占请求的 KV cache 应写入 storage，以便后续请求复用，减少重复计算开销。
- 资源预留调整：p 实例向 d 实例申请 block 时，d 实例需为运行中的请求预留 block ids，避免资源竞争导致的调度失败。
- 修复数据修改 bug：原 write_cache_to_storage 函数在构造 token_ids 时会意外修改 request 中的 prompt_token_ids，本 PR 通过引入中间变量 input_token_ids 修复此问题。

实现拆解

实现方案按模块拆解如下： | 模块 | 关键文件 | 主要变更 | |------|----------|----------| | 调度器 | fastdeploy/engine/sched/resource_manager_v1.py | 在 _trigger_preempt 函数中添加条件检查，当环境变量 FD_SAVE_OUTPUT_CACHE_FOR_PREEMPTED_REQUEST 开启且 storage backend 启用时，调用 write_cache_to_storage 或 write_cache_to_storage_decode；调整 preallocate_resource_in_d 函数使用 _get_can_schedule_prefill_threshold_block 计算总需 blocks。 | | 缓存管理器 | fastdeploy/cache_manager/prefix_cache_manager.py | 修改 can_allocate_gpu_blocks 函数，新增 try_free_gpu_blocks 参数（默认 True），在 request_match_blocks 中设为 False 以避免死锁；修复 write_cache_to_storage 和 write_cache_to_storage_decode 函数，防止修改原始 token_ids。 | | 环境配置 | fastdeploy/envs.py | 新增环境变量 FD_SAVE_OUTPUT_CACHE_FOR_PREEMPTED_REQUEST，默认值 1，控制是否在抢占时保存 cache 到 storage。 | | 缓存传输 | fastdeploy/cache_manager/cache_transfer_manager.py | 调整 read_storage_task 函数，在 token_ids 为空时传入 None，减少跨进程数据量。 | | 引擎 | fastdeploy/engine/common_engine.py | 将日志级别从 error 改为 warning，降低错误处理噪音。 |

评论区精华

Review 讨论中，Copilot 作为主要评论者，提出了多项关键洞察：

• 环境变量细节：

" 这里的注释里 preemted 拼写错误，建议更正为 preempted，避免后续搜索 / 文档引用时产生歧义。" " 这个环境变量是新引入的 ' 是否开启 ' 开关，但默认值设为 '1' 会导致默认启用写入 storage 的行为，可能带来额外 I/O 与延迟，并改变历史默认行为。"

• 设计风险：

" 在 schedule() 持有 self.lock 的情况下，这里同步调用 write_cache_to_storage*() 可能会把潜在的 I/O/ 等待放到调度锁里，导致调度线程长时间阻塞甚至影响并发。"

• 类型一致性：

"ReadStorageTask/CacheTask 的 token_ids 在类型标注里是 List[int] 且为必填字段，但这里在非 attention_store 后端时传入 None。建议统一传入空列表（[]）或把 CacheTask.token_ids 改为 Optional[List[int]]。"

• 测试覆盖：

" 新增的 'preempt 时写 cache 到 storage' 逻辑目前在单测中没有覆盖 ... 建议补充对应的单元测试，避免该关键路径在后续重构中回归。" 作者 juncaipeng 在部分评论中简要解释修改原因，如 " 避免可能的死锁卡住 " 和 " 减少跨进程传输的数据量 "，但未直接回应风险建议。

风险与影响

技术风险：

1. 性能瓶颈：调度锁内同步 I/O 操作可能阻塞线程，放大抢占路径的尾延迟，影响系统响应时间。
2. 运行时错误：token_ids 传入 None 违反类型约定，若后续代码无条件操作可能引发异常。
3. 行为变更：环境变量默认开启改变历史行为，未评估场景下可能引入不必要 I/O 开销。
4. 回归风险：缺少单元测试，Codecov 报告显示 patch 覆盖率仅 57.14286%，新增逻辑易在重构中失效。

影响评估：

- 正面影响：提升缓存复用率，减少抢占请求的重新计算，优化 PD Disaggregation 资源利用率。
- 负面影响：默认开启写 cache 可能增加存储 I/O 和延迟；未解决的设计风险可能降低系统并发性。
- 团队影响：需关注环境变量配置和类型约定，review 中未决问题提示后续开发需加强设计评审。

关联脉络

从同仓库近期历史 PR 分析，本 PR 与多个 KVCache 和 Scheduler 相关变更形成关联脉络：

- PR #6929：修复 KVCache 中 hash 边界比较 bug，共享缓存管理逻辑，显示前缀缓存计算的持续优化。
- PR #6992：新增中断请求端点，涉及 Scheduler 和 KVCache 资源管理，与本 PR 的抢占处理互补。
- PR #7046 和 #7075：关于 KVCache storage cache 的锁添加与回滚，突显缓存并发控制的复杂性，与本 PR 中锁内调用风险相呼应，提示团队在类似场景需谨慎权衡同步与异步设计。 整体上，这些 PR 共同推动 FastDeploy 在缓存持久化和调度优化方向的演进，本 PR 作为其中一环，强化了抢占场景下的缓存复用能力，但遗留的设计争议需在后续迭代中解决。