

PR #51227 完整报告

vllm-project/vllm

[Bugfix][KV Offload] Clean up resources after initialization failure

合并时间: 2026-08-06 19:42

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/51227>

执行摘要

- 一句话: 修复 KV offload 初始化失败时 mmap 与 tier 资源泄漏
- 推荐动作: 值得精读, 尤其是异常路径确定性清理的设计: 单 try/except 收敛多步资源创建、逆序 shutdown 保证依赖顺序、primary_tier is not None 区分两种落底清理方式、re-raise 保留原始异常。建议阅读时关注 primary_tier.shutdown() 内部对 mmap 的清理契约, 以及清理二次失败被日志吞掉的取舍; 由于本 PR 未补测试, 可结合后续补充失败分支测试的 PR 一起评估。

功能与动机

PR body 明确指出要防止 KV offloading 初始化失败时 /dev/shm/vllm_offload_*.mmap 与 tier 资源泄漏, 覆盖三类失败场景: primary tier 构造失败、后续 secondary tier 创建失败、worker 构造失败; 并声明仅处理普通 Python 初始化异常, OOM、SIGKILL、进程崩溃不在范围内。作者还专门核对 #51116 与 #51081 与本次改动无关, 避免重复提交。

实现拆解

实现按以下步骤拆解:

1. tiering 侧 get_manager() 统一清理 (vllm/v1/kv_offload/tiering/spec.py): 把原先 "创建 scheduler mmap → 创建 primary tier → 逐个创建 secondary tier → 构造 TieringOffloadingManager → 校验 store_threshold" 的串行逻辑整体收进一个 try/except。异常时先逆序 shutdown 已创建成功的 secondary_tiers (tier 之间存在依赖, 逆序保证后创建的先释放), 再 shutdown primary_tier (其内部会处理 scheduler mmap), 若 primary tier 尚未创建成功则直接调用 scheduler_mmap.cleanup(); 最后把 _scheduler_mmap 置回 None 并 re-raise 保留原始异常。
2. 配置校验提前: TieringOffloadingSpec.get_manager() 中 store_threshold >= 2 的 ValueError 被移到 try 块之前, 使非法配置在消耗任何 mmap 资源前就失败, 避免先分配再报错造成资源浪费。
3. 两个 spec 的 create_worker() 失败清理: tiering/spec.py 与 cpu/spec.py 的 create_worker() 都用 try/except 包裹 CUOffloadingWorker 构造, 构造失败时对 worker 侧的 SharedOffloadRegion 调用 cleanup() 后重抛, 防止 worker mmap 泄漏。
4. 测试与配套: 本 PR 没有新增测试; PR body 承认初始化失败清理分支未被现有单元测试覆盖, 现有 SharedOffloadRegion 测试只验证底层 mmap/fd/file 清理行为。测试计划仅为

pre-commit 与 Linux 单元测试回归。

关键文件：

- vllm/v1/kv_offload/tiering/spec.py (模块 分层卸载；类别 source；类型 core-logic；符号 get_manager, create_worker)：核心改动：get_manager() 将资源创建序列收敛为单个 try/except，失败时逆序 shutdown secondary tiers、shutdown primary tier 或直接 cleanup scheduler mmap，并保留原始异常；同时把 store_threshold 校验提前到分配之前，create_worker() 失败时清理 worker mmap。
- vllm/v1/kv_offload/cpu/spec.py (模块 CPU 卸载；类别 source；类型 core-logic；符号 create_worker)：create_worker() 在 CPUOffloadingWorker 构造失败时清理已创建的 SharedOffloadRegion mmap，防止 worker 侧 mmap 泄漏，与 tiering 侧改动配套。

关键符号：get_manager, create_worker

关键源码片段

vllm/v1/kv_offload/tiering/spec.py

核心改动：get_manager() 将资源创建序列收敛为单个 try/except，失败时逆序 shutdown secondary tiers、shutdown primary tier 或直接 cleanup scheduler mmap，并保留原始异常；同时把 store_threshold 校验提前到分配之前，create_worker() 失败时清理 worker mmap。

vllm/v1/kv_offload/tiering/spec.py 中 get_manager() 的异常清理 (PR #51227 合并后)

```
def get_manager(self) -> OffloadingManager:
```

```
    if not self._manager:
```

```
        # store_threshold 校验提前到资源分配之前：非法配置不应先消耗 mmap 资源
```

```
        if int(self.extra_config.get("store_threshold", 0)) >= 2:
```

```
            raise ValueError(
```

```
                "store_threshold is not supported for TieringOffloadingSpec"
```

```
            )
```

```
    scheduler_mmap: SharedOffloadRegion | None = None
```

```
    primary_tier: CPUPrimaryTierOffloadingManager | None = None
```

```
    secondary_tiers = []
```

```
    try:
```

```
        # scheduler 侧 mmap (rank=None)，primary tier 基于它 eager 创建
```

```
        # memoryview，因此必须先创建 scheduler mmap
```

```
        scheduler_mmap = SharedOffloadRegion(
```

```
            engine_id=self._engine_id,
```

```
            num_blocks=self.num_blocks,
```

```
            rank=None,
```

```
            kv_bytes_per_block=self.kv_bytes_per_chunk,
```

```
            cpu_page_size=self.cpu_page_size_per_worker,
```

```
        )
```

```
        self._scheduler_mmap = scheduler_mmap
```

```
    primary_tier = CPUPrimaryTierOffloadingManager(
```

```
        num_blocks=self.num_blocks,
```

```

        cache_policy=self.eviction_policy,
        cache_policy_module_path=self.cache_policy_module_path,
        enable_events=self.kv_events_config.enable_kv_cache_events,
        mmap_region=scheduler_mmap,
    )

    # 逐个创建 secondary tier, 任一步失败都进入统一清理
    primary_kv_view = primary_tier.get_kv_memoryview()
    for i, tier_config in enumerate(self.secondary_tier_configs):
        tier = SecondaryTierFactory.create_secondary_tier(
            tier_config, primary_kv_view, self
        )
        secondary_tiers.append(tier)
        logger.info("Created secondary tier #%d (%s)", i, tier.tier_type)

    tiering_manager = TieringOffloadingManager(
        primary_tier=primary_tier,
        secondary_tiers=secondary_tiers,
    )
    self._manager = tiering_manager
except Exception:
    # 逆序 shutdown 已创建成功的 secondary tier, 保证后创建的先释放
    for tier in reversed(secondary_tiers):
        try:
            tier.shutdown()
        except Exception:
            logger.exception(
                "Failed to shut down secondary tier during "
                "initialization cleanup"
            )
    # primary tier 的 shutdown 内部负责清理 scheduler mmap;
    # 若它尚未创建成功, 则直接落底清理 mmap
    if primary_tier is not None:
        try:
            primary_tier.shutdown()
        except Exception:
            logger.exception(
                "Failed to shut down primary tier during "
                "initialization cleanup"
            )
    elif scheduler_mmap is not None:
        try:
            scheduler_mmap.cleanup()
        except Exception:
            logger.exception(
                "Failed to clean up scheduler mmap during "
                "initialization cleanup"
            )
    self._scheduler_mmap = None

```

raise # 保留原始异常，让上层看到真正的失败原因

return self._manager

vllm/v1/kv_offload/cpu/spec.py

create_worker() 在 CPUOffloadingWorker 构造失败时清理已创建的 SharedOffloadRegion mmap，防止 worker 侧 mmap 泄漏，与 tiering 侧改动配套。

vllm/v1/kv_offload/cpu/spec.py 中 create_worker() 的清理逻辑 (PR #51227 合并后)

```
def create_worker(self, kv_caches: CanonicalKVCaches) -> CPUOffloadingWorker:
    mmap_region: SharedOffloadRegion | None = None
    # num_blocks == 0 时 mmap 区域大小为 0 无法映射，回退到 tensor 路径 (空 tensor)
    if self._uses_shared_region() and self.num_blocks > 0:
        # replicated 布局下所有 rank 共用 slot 0 (单个 MLA 副本) ;
        # 否则每个 rank 按物理设备索引取自己的 slot
        if self.replicated_layout:
            rank = 0
        else:
            world_size = self.config.parallel.world_size
            rank = torch.accelerator.current_device_index() % world_size
        mmap_region = SharedOffloadRegion(
            engine_id=self.config.engine_id,
            num_blocks=self.num_blocks,
            rank=rank,
            kv_bytes_per_block=self.kv_bytes_per_chunk,
            cpu_page_size=self.cpu_page_size_per_worker,
        )
    try:
        return CPUOffloadingWorker(
            kv_caches=kv_caches,
            blocks_per_chunk=self.blocks_per_chunk,
            num_cpu_blocks=self.num_blocks,
            mmap_region=mmap_region,
        )
    except Exception:
        # worker 构造失败时主动清理已创建的 mmap，避免 /dev/shm 文件残留
        if mmap_region is not None:
            mmap_region.cleanup()
        raise
```

评论区精华

核心讨论围绕 get_manager() 的清理结构展开。orozery 转述 Claude 的 review 建议：应使用单个 try/except 包裹整个分配序列 (primary tier + secondary tiers + TieringOffloadingManager 构造)，用 primary_tier is not None 判断调用 primary_tier.shutdown() (内部处理 mmap) 还是 scheduler_mmap.cleanup()，这样更简洁且能覆盖原 PR 漏掉的 TieringOffloadingManager 构造失败场景。作者采纳后询问是否引入 contextlib.ExitStack 简化错误处理，orozery 表示先不用它、看看效果。此外，orozery 在

inline review 中指出 re-raise 前的 logger.exception 日志冗余，要求删除，作者在最后一个提交 rm log 中落实。

- get_manager 清理合并为单个 try/except (design): 作者重写为单 try/except 结构，并额外把 store_threshold 校验提前到分配之前。
- 是否引入 contextlib.ExitStack (design): orozery 表示先不用 ExitStack，看单 try/except 的实际效果；最终未引入。
- 删除 re-raise 前的冗余错误日志 (style): 作者在最后一个提交 ae8524be (rm log) 中删除。

风险与影响

- 风险：
 - 缺少测试覆盖：PR body 明确承认初始化失败清理分支没有直接的单元测试，现有 SharedOffloadRegion 测试只覆盖底层 mmap/fd/file 清理行为；误删或漏删资源的回归只能在集成环境暴露。
 - 清理逻辑的二次失败被吞掉：tier.shutdown()、primary_tier.shutdown()、scheduler_mmap.cleanup() 各自包裹 try/except，清理失败只记日志，与 " 防泄漏 " 目标存在取舍：原始异常必须优先传播。
 - 依赖 shutdown 语义：primary_tier.shutdown() 需要内部负责 scheduler mmap 清理，若后续重构改变 shutdown 职责，会出现 mmap 残留。
 - 行为变化：store_threshold 校验提前属于可见的顺序变化，但方向更合理；正常路径无行为变化。
 - 影响：影响范围集中在 vllm v1 引擎启用 KV offload / tiering 配置的初始化失败路径。对用户而言，修复避免了反复重试初始化时 /dev/shm 被残留 mmap 文件耗尽的问题，属于可靠性改进；对系统而言，失败路径的资源生命周期现在与成功路径一致，减少多副本 (DP replica) 场景下的资源碎片。对团队的后续影响是确立了 " 单 try/except + 逆序清理 + 保留原始异常 " 的失败清理模式，可复用于其他多资源初始化点。
- 风险标记：缺少测试覆盖，异常路径清理，依赖 shutdown 内部清理 mmap

关联脉络

- PR #51116 [Bugfix][KV Offload] Fall back when MADV_POPULATE_WRITE is unsupported: 同属 vllm/v1/kv_offload 模块的 mmap 资源生命周期修复；PR body 专门说明与其无关 (mmap fallback)，但二者共同完善 KV offload 的 mmap 健壮性。