PR #50358 完整报告

vllm-project/vllm

[Bugfix] Fail fast with a clear error when CPU offload region exceeds available space

合并时间：2026-08-06 01:59

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/50358

执行摘要

- 一句话：CPU KV 卸载超容时快速报错并清理，避免 /dev/shm 隐式失败
- 推荐动作：建议精读。虽然改动量很小，但 review 中关于 try/except 范围、错误传播与并发清理语义的讨论质量很高，值得 KV offload 相关维护者参考。重点关注：容量检查为何只放在 creator 路径、为何 except 不应扩大到 mmap/madvise、以及如何避免 0 字节 stub 拖垮 joiners。测试写法（直接 monkeypatch helper）也值得借鉴。

功能与动机

issue #46949 明确指出：用户设置过高的 cpu_bytes_to_use（或 --kv-offloading-size）时，vLLM 会在 vllm/v1/kv_offload/cpu/shared_offload_region.py 的 mmap_obj.madvise 处抛出无上下文的 OSError: [Errno 14] Bad address，用户完全无法判断是容量问题。PR body 进一步说明，#50094 之后默认 CPUOffloadingSpec 在 CUDA/ROCm 上使用 SharedOffloadRegion，因此两个配置入口（CLI flag 与 kv_connector_extra_config）都需要在错误消息中被点名，并给出可操作的建议（如 --shm-size 或 --ipc=host）。

实现拆解

实现拆解

该 PR 仅改动了 2 个文件：源码 vllm/v1/kv_offload/cpu/shared_offload_region.py 与配套测试 tests/v1/kv_offload/cpu/test_shared_offload_region.py。

1. 重构 SharedOffloadRegion.__init__ 的 try/except 结构：原有代码把 os.open(O_EXCL) 和 os.ftruncate 混在同一个 try 块中，FileExistsError 分支同时承担 joiner 重开文件与等待逻辑。新结构将 FileExistsError 走 joiner 路径、else 分支走 creator 路径，两条路径的错误处理互不纠缠。
2. Creator 路径增加容量预检与失败清理：creator 赢得 O_EXCL 后调用 check_shm_free_space(self.total_size_bytes)（复用 vllm.distributed.device_communicators.shm_broadcast 已有 helper），若检查或 os.ftruncate 失败，先 os.unlink(self.mmap_path) 再 os.close(self.fd)，最后原样抛出异常。这样并发 joiners 不会落在一个 0 字节 stub 上于 _wait_for_file_size 中空转 30 秒。
3. Joiner 路径关闭 fd 防泄漏：_wait_for_file_size 抛出的 TimeoutError 或 OSError 时先 os.close(self.fd) 再重新抛出；成功打开既有文件的行为与原先一致。

4. 测试配套：新增 test_insufficient_space_raises_clear_error（monkeypatch check_shm_free_space 抛 RuntimeError，验证异常透传并断言 unlink/close 各调用一次）与 test_ftruncate_failure_cleans_up_creator（monkeypatch ftruncate 抛 OSError，验证同样清理）。测试直接通过公共 API 构造 SharedOffloadRegion，避免依赖真实 /dev/shm 容量。

变更整体只影响启动期的失败路径；正常容量下 creator/joiner 行为与之前一致。

关键文件：

- vllm/v1/kv_offload/cpu/shared_offload_region.py（模块 共享卸载区；类别 source；类型 core-logic；符号 SharedOffloadRegion.init, _wait_for_file_size）：核心源码文件。重构 SharedOffloadRegion.__init__的 creator/joiner 路径，在 creator 赢得 O_EXCL 后调用 check_shm_free_space 做容量预检，失败时清理 0 字节 stub 并关闭 fd，是修复 issue #46949 的主逻辑。
- tests/v1/kv_offload/cpu/test_shared_offload_region.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_insufficient_space_raises_clear_error, test_ftruncate_failure_cleans_up_creator）：测试配套。新增两个测试用例验证容量不足与 ftruncate 失败时异常能透传且 creator 能正确清理 mmap 文件与 fd，是防止该修复回归的关键保障。

关键符号：SharedOffloadRegion.init, _wait_for_file_size, test_insufficient_space_raises_clear_error, test_ftruncate_failure_cleans_up_creator

关键源码片段

vllm/v1/kv_offload/cpu/shared_offload_region.py

核心源码文件。重构 SharedOffloadRegion.__init__的 creator/joiner 路径，在 creator 赢得 O_EXCL 后调用 check_shm_free_space 做容量预检，失败时清理 0 字节 stub 并关闭 fd，是修复 issue #46949 的主逻辑。

```
try:
    # 独占创建：只有第一个 worker 成功，成为 creator
    self.fd = os.open(
        self.mmap_path, os.O_CREAT | os.O_EXCL | os.O_RDWR, 0o600
    )
except FileExistsError:
    # Joiner 路径：另一个 worker 已赢得 O_EXCL，直接打开既有文件
    # 并等待其被 truncate 到预期大小；等待失败时关闭 fd 再抛出。
    self.fd = os.open(self.mmap_path, os.O_RDWR)
    try:
        _wait_for_file_size(self.fd, self.total_size_bytes)
    except (TimeoutError, OSError):
        os.close(self.fd) # 避免 fd 泄漏
        raise
    logger.info("Opened existing mmap file %s", self.mmap_path)
else:
    # Creator 路径：只有它需要预留容量，因此只在 O_EXCL 获胜后
```

```
    # 才调用 check_shm_free_space，避免 joiners 看到被 creator 消耗的
    # tmpfs 空闲空间后误报 Insufficient-space。
    try:
        check_shm_free_space(self.total_size_bytes)
        os.ftruncate(self.fd, self.total_size_bytes)
    except (RuntimeError, OSError):
        # 清理 0 字节 stub，防止并发 joiners 在 _wait_for_file_size 中
        # 空转满 30 秒超时；错误原样抛出以保留可操作提示。
        os.unlink(self.mmap_path)
        os.close(self.fd)
        raise
    self._creator = True
    logger.info(
        "Created mmap file %s (%.2f GB)",
        self.mmap_path,
        self.total_size_bytes / 1e9,
    )
```

上述片段展示了本 PR 的核心：容量检查只发生在 creator 路径，且失败的清理动作（unlink + close）与原异常透传绑定在同一窄 except 中。

tests/v1/kv_offload/cpu/test_shared_offload_region.py

测试配套。新增两个测试用例验证容量不足与 ftruncate 失败时异常能透传且 creator 能正确清理 mmap 文件与 fd，是防止该修复回归的关键保障。

```
def test_insufficient_space_raises_clear_error(monkeypatch):
    """Creator 容量不足时必须清理 stub 并透传清晰错误。"""
    import vllm.v1.kv_offload.cpu.shared_offload_region as region

    engine_id = str(uuid.uuid4())
    mmap_path = f"/dev/shm/vllm_offload_{engine_id}.mmap"
    mock_open = MagicMock(return_value=9999)
    mock_unlink = MagicMock()
    mock_close = MagicMock()
    monkeypatch.setattr(region.os, "open", mock_open)
    monkeypatch.setattr(region.os, "unlink", mock_unlink)
    monkeypatch.setattr(region.os, "close", mock_close)
    # 直接让容量检查抛错，验证异常透传与清理动作
    monkeypatch.setattr(
        region,
        "check_shm_free_space",
        lambda *a, **kw: (_ for _ in ()).throw(
            RuntimeError("Insufficient space in /dev/shm: 30 GB required.")
        ),
    )

    with pytest.raises(RuntimeError, match="Insufficient space"):
        SharedOffloadRegion(
            engine_id=engine_id,
```

```
            num_blocks=4,
            rank=0,
            kv_bytes_per_block=PAGE_SIZE,
            cpu_page_size=PAGE_SIZE,
        )

    # 必须同时 unlink 0 字节 stub 并关闭 fd
    mock_unlink.assert_called_once_with(mmap_path)
    mock_close.assert_called_once_with(9999)
```

该测试通过 monkeypatch 直接验证两条核心契约：错误信息透传与清理动作的原子性。

评论区精华

评论区精华

- 容量检查位置：orozery 对早期版本中 if not os.path.exists(...) 的 guard 表示质疑（"I don't understand why we need this check"）。作者最初解释是为了避免 joiners 在 creator 的 madvise 已消耗 tmpfs 空闲后误报容量不足（TOCTOU）。最终 orozery 建议把检查移到 os.open(O_EXCL) 之后只对 creator 执行，作者采纳，guard 被移除，逻辑更简洁且无竞态。
- except 范围收窄：作者提出把 mmap.mmap、madvise、torch.frombuffer 也纳入错误捕获并统一清理。orozery 的 Claude agent 给出三点反对：unlink 的唯一目的是保护 joiners 不因 0 字节 stub 挂起，而这只会发生在 ftruncate 失败时；mmap 失败时文件尺寸已经正确，joiners 不会挂起；把 joiner 路径（FileExistsError）与清理路径（OSError）混在同一个 try 里脆弱，因为 OSError 是 FileExistsError 的父类，异常顺序稍错就会误捕。最终 except 被收窄为只包裹 check_shm_free_space + ftruncate。
- 保留 helper 原始错误：orozery 指出 check_shm_free_space 已经给出可操作消息（含请求 / 可用容量与调整建议），不应再改写异常。commit "Preserve helper capacity errors" 落实了这一点。
- 测试收敛：orozery 建议删除 test_wait_for_file_size_fails_when_file_is_unlinked（罕见 unlinked-inode 路径，无此检查也只会回退到 30 秒超时），最终测试聚焦于容量校验和 creator 清理两条实际路径。
- 容量检查位置：exists guard 还是 O_EXCL 之后 (design): 采纳 orozery 建议：容量检查只放在 creator 路径（O_EXCL 获胜后），移除 exists guard。
- except 清理范围是否应覆盖 mmap/madvise/torch.frombuffer (correctness): except 收窄为只包裹 check_shm_free_space + ftruncate，清理动作（unlink + close）后原样抛出。
- 是否保留 check_shm_free_space 原始错误消息 (design): 保留 helper 原始错误，不做异常改写。
- 罕见 unlinked-inode 路径测试是否保留 (testing): 删除该测试，测试聚焦容量校验与 creator 清理。
- joiner 失败清理的异常类型收窄 (style): 采用简化写法，避免过度捕获。

风险与影响

- 风险：### 风险分析
- TOCTOU 竞态：check_shm_free_space 与 ftruncate/madvise 之间存在时间窗口，其他进程可能在这期间占满 /dev/shm，极端情况下 madvise 仍可能抛出 Bad address。该窗口与原有行为相比并未扩大，且概率低。
- Creator 清理与 joiner 竞态：若 joiner 在 os.unlink 之后才执行 os.open，会得到 FileNotFoundError 并快速失败（而非挂起），行为可接受；若 joiner 在 unlink 前已打开文件，POSIX 语义下 inode 仍有效，后续行为正常。
- 对非 Linux 平台的隐式依赖：check_shm_free_space 内部对 /dev/shm 是否存在有短路处理（从早期测试版本可见 os.path.isdir 分支），但本 PR 未显式验证 macOS/Windows 等无 /dev/shm 平台的完整行为，存在一定不确定性。
- 影响面：改动集中启动期失败路径，正常路径（creator 容量充足、joiner 等待）行为不变，回归风险低。
- 影响：### 影响分析
- 用户侧：配置过大的 --kv-offloading-size 或 cpu_bytes_to_use 时，启动阶段立即得到包含请求容量、可用容量和调整建议（如 --shm-size、--ipc=host）的错误消息，不再看到晦涩的 OSError: [Errno 14] Bad address。
- 系统侧：失败时及时清理 0 字节 mmap 文件，避免并发 worker 在 _wait_for_file_size 中空转 30 秒后超时崩溃，缩短多卡场景下的失败感知时间。
- 团队侧：修复 issue #46949，并统一了与 #47073、#46959 两个停滞重复 PR 的实现口径，最终采用复用现有 helper + creator 路径检查的保守方案。对 KV offload 功能线（近期 #50992、#51007、#50321 等）提供了更健壮的启动期保护。
- 风险标记：启动路径变更，/dev/shm 容量 TOCTOU 竞态，并发清理与 joiner 竞态，错误透传依赖外部 helper

关联脉络

- PR #50094 Default CPUOffloadingSpec uses SharedOffloadRegion on CUDA/ROCm（PR body 提及）：PR body 明确指出 #50094 之后默认 CPUOffloadingSpec 在 CUDA/ROCm 上使用 SharedOffloadRegion，这正是本次容量检查影响面扩大的直接原因。
- PR #50992 [Perf][KV Offload] Avoid quadratic ARC batch eviction: 同属 vllm/v1/kv_offload CPU 卸载子系统，本 PR 的容量预检为这类 CPU 卸载路径提供启动期保护。
- PR #51007 [KV Offload] Support out-of-tree secondary tier managers via module_path: 同属 kv_offload 功能线，tiering 与 CPU 策略持续演进，容量与资源管理是共同关注点。
- PR #50321 [KV Offload] Support partial secondary-tier load results: 同属 kv_offload 加载与容量管理路径，本 PR 的错误信息透传与清理逻辑与其互补。
- PR #47073 Duplicate PR addressing same issue（stalled，PR body 提及）：PR body 说明 #47073 与 #46959 针对同一问题但实现停滞，本 PR 最终采用复用现有 helper + creator 路径放置的方案。