PR #52514 完整报告

vllm-project/vllm

[Core] Add CuMemAllocator.discard() for tag-selective GPU memory release

合并时间：2026-08-16 23:15

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/52514

执行摘要

- 一句话：为 CUDA/XPU 分配器新增 tag 选择性 discard()，细化 sleep/wake_up 状态机
- 推荐动作：值得精读。该 PR 展示了“内存分配器状态机”这一类基础设施的典型设计手法：用显式状态位（is_asleep）区分已 unmap 与仍 mapped，用幂等告警而非静默变更来应对策略冲突，并在 unmap 前同步设备保证正确性。重点留意 discard() 后访问未唤醒分配的非法地址风险、以及默认 offload_tags=('default',) 的调用陷阱。若要在产品中集成，建议补充策略冲突分支的单元测试并考虑将多次全设备同步合并为一次。

功能与动机

PR body 明确这是 #46438 的 follow-up：PR #46438 introduces tag-selective discard(), allowing stale allocations such as KV cache to be released while keeping model weights mapped and usable，目标场景是 multi-model and RL weight-sync scenarios where stale data (e.g. a previous model's KV cache) should be freed to make room for new allocations while the current weights remain usable。原有 sleep() 是全量操作，无法只释放部分 tag，且 asleep 分配已被 unmap 后无法再被正确恢复，因此本 PR 引入显式 is_asleep 状态与策略冲突检测，保证重复请求幂等、冲突请求不改动既定策略。

实现拆解

在 vllm/device_allocator/init.py 的 MemAllocator Protocol 中新增 discard(tags) 抽象方法，与 sleep()/wake_up() 并列，要求 CuMemAllocator（CUDA/ROCm）与 XpuMemAllocator（XPU）同时实现，保证上层调用方无需感知平台差异。

两个实现里 AllocationData 原本只有 cpu_backup_tensor 可用于区分“是否 offload 过”，但无法区分“已 unmap（asleep）”与“仍 mapped”。本 PR 在 sleep() 中对每个被 unmap 的分配显式置 data.is_asleep = True；对已 asleep 的分配不再重复 unmap，而是计算 requests_offload（本次请求是否要 offload）与 was_offloaded（是否有 CPU 备份）是否一致，不一致时置 has_policy_conflict 并最终打 warning，不改动既定策略。这一步直接支撑了 body 里的论证：discard 过的分配没有 CPU 备份，后续 offload 请求无法恢复原始内容；offload 过的分配已有备份，后续 discard 不应静默丢弃备份。

cumem.py 与 xpumem.py 同时新增 discard()：遍历 pointer_to_data，跳过 tag 不匹配的分配；对匹配且仍 mapped 的分配，先 torch.accelerator.synchronize(device) 等待设备侧 kernel 完成，再 unmap_and_release() 释放物理内存并置 is_asleep；对已 asleep 且有

CPU 备份的分配只告警（策略冲突）。与 sleep() 的差异是仅触碰指定 tag，其余分配保持 mapped 可直接使用。

两个实现都在遍历开头加 if not data.is_asleep: continue，避免对从未 sleep 的分配重复 create_and_map/create_and_allocate；remap后清is_asleep，有CPU备份的再拷回数据。这修复了此前 wake_up() 可能对 mapped 状态分配重复映射的隐患。

tests/basic_correctness/test_mem.py 新增 test_discard_tags：分别用 weights 与 kv_cache 两个 tag 申请内存，discard('kv_cache') 后断言空闲显存增加且 weights 内容仍有效，wake_up() 后 kv 的 VA 重新映射（内容不保留），最后完整 sleep(offload_tags='weights')/wake_up() 周期仍通过。提交历史中特别修正了测试里未显式传 offload_tags 的问题：默认值是 ('default',)，省略会把 weights 当 discard 处理，唤醒后内容清零导致断言失败。

关键文件：

- vllm/device_allocator/cumem.py（模块 内存分配器；类别 source；类型 core-logic；符号 discard, sleep, wake_up）：核心实现之一：新增 CuMemAllocator.discard()，sleep() 增加策略冲突检测与 is_asleep 状态维护，wake_up() 只重映射 asleep 分配；PR 标题即指向该文件。
- vllm/device_allocator/xpumem.py（模块 内存分配器；类别 source；类型 core-logic；符号 discard, sleep, wake_up）：XPU 并行实现，逻辑与 cumem.py 对齐（使用 torch.accelerator.synchronize 与 _xpu_memcpy_sync），保证双平台行为一致；inline 评论也落在此文件。
- tests/basic_correctness/test_mem.py（模块 内存测试；类别 test；类型 test-coverage；符号 test_discard_tags）：新增 test_discard_tags 覆盖按 tag 选择性释放、权重保持可用与 sleep/wake_up 组合周期；测试中还暴露并修复了默认 offload_tags 的陷阱。
- vllm/device_allocator/__init__.py（模块 内存分配器；类别 source；类型 interface；符号 discard）：MemAllocator Protocol 新增 discard() 抽象方法，统一 CUDA/XPU 接口契约，是所有调用方与实现方的公共入口。

关键符号：CuMemAllocator.discard, XpuMemAllocator.discard, CuMemAllocator.sleep, XpuMemAllocator.sleep, CuMemAllocator.wake_up, XpuMemAllocator.wake_up, MemAllocator.discard, test_discard_tags

关键源码片段

vllm/device_allocator/cumem.py

核心实现之一：新增 CuMemAllocator.discard()，sleep() 增加策略冲突检测与 is_asleep 状态维护，wake_up() 只重映射 asleep 分配；PR 标题即指向该文件。

```
def discard(self, tags: tuple[str, ...] | str) -> None:
    """按 tag 选择性释放 GPU 物理内存，不先备份到 CPU。

    与 sleep() 不同，discard() 只处理匹配 tags 的分配，其余分配
    保持 mapped 状态，可继续被模型权重等使用。
    """
```

```
    if isinstance(tags, str):
        tags = (tags,)

    discarded_bytes = 0
    has_policy_conflict = False
    for data in self.pointer_to_data.values():
        if data.tag not in tags:
            continue
        if data.is_asleep:
            # 已经 asleep 的分配若之前被 offload（有 CPU 备份），
            # 再 discard 会静默丢弃备份、改变既有策略，故只告警不改动。
            if data.cpu_backup_tensor is not None:
                has_policy_conflict = True
            continue
        # unmap 前同步设备：防止仍在运行的 kernel 访问已释放物理页，
        # 否则会触发 CUDA_ERROR_ILLEGAL_ADDRESS。
        torch.accelerator.synchronize(data.handle[0])
        unmap_and_release(data.handle)
        data.is_asleep = True
        discarded_bytes += data.handle[1]

    logger.info(
        "CuMemAllocator: discarded %.2f GiB for tags %s.",
        discarded_bytes / 1024**3,
        tags,
    )

    if has_policy_conflict:
        logger.warning(
            "CuMemAllocator: discard cannot change the policy of "
            "already-asleep allocations; the existing policy was kept."
        )
```

tests/basic_correctness/test_mem.py

新增 test_discard_tags 覆盖按 tag 选择性释放、权重保持可用与 sleep/wake_up 组合周期；测试中还暴露并修复了默认 offload_tags 的陷阱。

```
@create_new_process_for_each_test("fork" if current_platform.is_cuda() else "spawn")
def test_discard_tags():
    """验证 discard(tags) 只释放指定 tag 的显存，其他 tag 仍可用。"""
    allocator = get_mem_allocator_instance()

    with allocator.use_memory_pool("weights"):
        weights = torch.ones(1024, 1024, device=DEVICE_TYPE)

    with allocator.use_memory_pool("kv_cache"):
        kv = torch.ones(512, 512, device=DEVICE_TYPE)

    free_bytes = torch.accelerator.get_memory_info()[0]
```

```
# 只 discard kv_cache，weights 应保持 mapped 且内容有效
allocator.discard("kv_cache")

free_bytes_after_discard = torch.accelerator.get_memory_info()[0]
assert free_bytes_after_discard > free_bytes

# 权重仍然可用
assert torch.allclose(weights, torch.ones_like(weights))

# wake_up 后 kv_cache 的 VA 被重新映射，内容不保留但分配有效
allocator.wake_up()
assert kv.shape == (512, 512)

# discard 之后完整的 sleep/wake_up 周期仍然工作：
# 注意这里必须显式传 offload_tags="weights"，因为默认值是
# ('default',)，若省略则 weights 会被当作 discard 处理，
# 唤醒后内容为零，allclose 断言会失败。
allocator.sleep(offload_tags="weights")
allocator.wake_up()
assert torch.allclose(weights, torch.ones_like(weights))
```

评论区精华

- aoshen02 在 xpumem.py 的 sleep() 新增 docstring 上留下一句评论：“bit verbose”，这是本 PR 唯一的 inline review 评论；最终通过最后一次提交 “simplify doc” 收敛文档表述，而后 aoshen02 与 ZJY0516 先后批准。
- PR body 对策略冲突的论证是全 PR 的核心设计讨论：“A discarded allocation has no CPU backup, so a later offload request cannot recover and back up its original contents. An offloaded allocation already has a CPU backup, so a later discard request cannot silently change the established policy. Repeated requests using the same policy remain silent and idempotent.” 代码中的 has_policy_conflict 告警正是这一语义的落地。
- xpumem.py 新 docstring 过于冗长 (style): 通过最后一次提交 “simplify doc” 简化文档表述，随后 aoshen02 与 ZJY0516 批准合并。
- sleep/discard 策略冲突的幂等语义 (design): 以 has_policy_conflict 标志 + warning 方式落地，既不改变已 asleep 分配的策略，也避免静默的数据丢失；该设计贯穿 sleep() 与 discard()。

风险与影响

- 风险：
  - 非法地址访问风险：discard() 后 tensor 的虚拟地址仍被 Python 引用，但物理页已释放；在 wake_up() 之前访问该 tensor 会触发 CUDA_ERROR_ILLEGAL_ADDRESS。测试仅验证 shape，未验证内容，调用方必须自行保证丢弃后不再读取。这是该 API 最主要的误用风险。

- wake_up() 语义收紧的兼容性：此前 wake_up() 会对所有分配（含仍 mapped 的）执行 create_and_map()；现在只 remap is_asleep 的分配。若外部调用方依赖旧行为（例如在未 sleep 时调用 wake_up 强制重映射），行为将改变，不过新语义更正确。
- 同步开销：discard() 对每个匹配分配调用一次 torch.accelerator.synchronize()，涉及多个分配时会产生多次全设备同步；批量场景可考虑只同步一次，当前实现偏保守但安全。
- 策略冲突路径无测试覆盖：has_policy_conflict 的告警分支（重复 discard、discard 已 offload 的分配、sleep 改变已 asleep 分配策略）没有对应测试，行为依赖代码审查保证。
- 平台覆盖：XPU 路径与 CUDA 路径逻辑几乎一致，但测试主要在 CUDA 上验证；XPU 的 torch.accelerator.synchronize 与 _xpu_memcpy_sync 组合的真实行为仍需 CI 验证。
- 默认 tag 陷阱：sleep() 默认 offload_tags=('default',)，使用自定义 tag 的调用方若省略参数，分配会被当作 discard 处理——测试中已踩过并修复，属于 API 使用上的坑。

- 影响：
  - 用户 / 场景影响：启用 sleep-mode 分配器的部署（主要是 CUDA graph + 显存复用场景）获得新的显存精细控制手段。多模型串行部署与 RL 权重同步流程可以在不打断权重 mapped 状态的前提下回收陈旧 KV cache，减少显存碎片与不必要的全量 offload。
  - 系统影响：改动集中在 vllm/device_allocator（CUDA/ROCm 与 XPU 双平台），MemAllocator Protocol 增加方法属于向后兼容的接口扩展；sleep()/wake_up() 行为收紧对既有 sleep-mode 调用方有一定适配成本。
  - 团队影响：需要确认 vLLM 内部当前的 sleep() 调用点（如与 cudagraph、KV cache offload 相关的路径）不依赖旧 wake_up() 的全量重映射行为。
  - 风险标记：discard 后访问未唤醒内存会触发非法地址，wake_up 语义收紧可能影响既有调用方，策略冲突分支缺少测试覆盖，默认 offload_tags 存在使用陷阱，多次全设备同步有性能开销

关联脉络

- PR #46438 前置 PR（标题未在上下文中提供）：PR body 明确声明本 PR 是其 follow-up：46438 引入 tag-selective discard()，允许释放陈旧 KV cache 同时保持模型权重 mapped 可用；本 PR 在此基础上将 discard 落到 CUDA/XPU 两个分配器并细化 sleep/wake_up 交互与状态机。
- PR #43107 [Core] Check for GPU<->CPU syncs during CI: 双方都围绕 GPU 与 CPU 同步纪律：43107 在 CI 中检测并清理非必要同步点，本 PR 则要求在 unmap 前显式同步设备以避免非法地址错误，体现同一性能 / 正确性权衡主线。