PR #38503 完整报告

vllm-project/vllm

[ROCm][Engine] Fix GPU memory leaks in engine shutdown and test workaround for async KV prefix cache reset

合并时间：2026-04-25 13:25

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/38503

执行摘要

- 一句话：修复引擎关闭时 GPU 内存泄漏并添加诊断测试
- 推荐动作：建议引擎、内存管理相关开发者精读此 PR，重点了解：1）bound method 作为 LRU 缓存键导致的内存泄漏模式及解包方案；2）gc.freeze/unfreeze 的正确配对使用；3）异步资源传输与同步点设计的权衡。commit 历史展示了调试和返工过程，对理解设计演变有帮助。

功能与动机

当在进程内模式（VLLM_ENABLE_V1_MULTIPROCESSING=0）下运行时，引擎关闭后 GPU 内存未被释放，导致后续操作或测试出现 OOM。PR body 指出三个原因：1）supports_kw 的 @lru_cache 绑定 bound method 导致模型实例和 GPU 张量被强引用；2）EngineCore.__init__ 调用了 gc.freeze() 但 shutdown() 从未调用 gc.unfreeze()，使引擎对象对 GC 不可见；3）GPUWorker.shutdown() 未释放模型权重、KV 缓存、旋转嵌入缓存、编译上下文和工作空间单例。此外，在 KV offloading 场景下，reset_prefix_cache 可能因异步传输未完成而无限返回 False，测试需要 workaround。

实现拆解

1. 修复 supports_kw LRU 缓存泄漏

- 文件：vllm/utils/func_utils.py
- 将原有带 @lru_cache 的 supports_kw 重命名为 _supports_kw，并创建新的非缓存版本 supports_kw 作为外观。
- 在新 supports_kw 中，通过 hasattr(callable, '__func__') 判断是否为 bound method，若是则解包为 callable.__func__ 再调用 _supports_kw，避免以实例 method 为缓存键，防止模型实例被长期引用。

2. 解除 gc.freeze()

- 文件：vllm/v1/engine/core.py
- 在 EngineCore.shutdown() 方法末尾添加 gc.unfreeze()，与 __init__ 中的 gc.freeze() 配对，使引擎中分配的对象（模型权重、KV 缓存等）对 GC 重新可见。

3. 添加显式 GPU 资源释放

- 文件：vllm/v1/worker/gpu_model_runner.py & vllm/v1/worker/gpu_worker.py
- 在 GPUModelRunner 中新增 shutdown() 方法：调用 _cleanup_profiling_kv_cache()（含 torch.accelerator.synchronize()）、清空 static_forward_context、释放模型 self.model = None、清空旋转嵌入缓存 _ROPE_DICT.clear()、重置工作空间 reset_workspace_manager()。
- 在 GPUWorker.shutdown() 中，添加对 self.model_runner.shutdown() 的调用。

4. 添加 GPU 内存泄漏诊断测试

- 新增文件：tests/v1/kv_connector/unit/test_rixl_gpu_mem_diag.py
- 包含 test_gpu_memory_rixl_hma 和 test_gpu_memory_no_rixl_baseline 两个测试，分别在带与不带 NixlConnector 的场景下，跟踪引擎创建 / 推理 / 关闭后的 GPU 内存是否回到基线。

5. 修复异步 KV 前缀缓存重置的测试 workaround

- 文件：tests/v1/kv_offload/test_cpu_offloading.py
- 修改 _wait_for_prefix_cache_reset，在重试 reset_prefix_cache 之前发送一个 dummy single-token prefill（llm.generate([TokensPrompt(prompt_token_ids=[0])], ...)）强制引擎步进，以排空异步 offload 传输并释放 GPU 块。

6. 升级 UCX 版本修复 RIXL 内存泄漏

- 文件：docker/Dockerfile.rocm
- 将 UCX 分支更新到 33b3b2a2，该版本包含与 NIXL 一起使用时的内存泄漏修复。

关键文件：

- tests/v1/kv_connector/unit/test_rixl_gpu_mem_diag.py（模块 内存诊断；类别 test；类型 test-coverage；符号 _mb, _gpu_snapshot, _full_gpu_cleanup, test_gpu_memory_rixl_hma）：新增诊断测试，专门验证 GPU 内存在引擎 shutdown 后完全释放，针对 RIXL/NixlConnector 场景，是回归测试的关键。
- vllm/utils/func_utils.py（模块 工具函数；类别 source；类型 core-logic；符号 supports_kw, _supports_kw）：通过将 supports_kw 拆分为非缓存外观和缓存内部函数，在缓存键中解包 bound method，根除因 LRU 缓存绑定实例导致的内存泄漏。
- vllm/v1/worker/gpu_model_runner.py（模块 模型运行器；类别 source；类型 core-logic；符号 shutdown）：新增 shutdown() 方法，集中释放 KV 缓存、模型权重、旋转嵌入缓存和工作空间，是 GPU 内存清理的核心。
- vllm/v1/engine/core.py（模块 引擎核心；类别 source；类型 dependency-wiring）：在 EngineCore.shutdown() 中添加 gc.unfreeze()，与 __init__ 中的 gc.freeze() 配对，使得引擎对象在关闭后对 GC 可见。
- vllm/v1/worker/gpu_worker.py（模块 GPU 工作器；类别 source；类型 core-logic）：在 GPUWorker.shutdown() 中新增调用 model_runner.shutdown()，串联模型运行器的清理逻辑。
- tests/v1/kv_offload/test_cpu_offloading.py（模块 CPU 卸载；类别 test；类型 test-coverage）：修改 _wait_for_prefix_cache_reset，在重试前发送 dummy token 以强

制引擎步进，避免 reset_prefix_cache 因异步传输未完成而卡住。

- docker/Dockerfile.rocm（模块 容器构建；类别 infra；类型 infrastructure）：将 UCX 分支更新到包含内存泄漏修复的版本，解决 NIXL 在 ROCm 上使用 UCX 时的 GPU 内存泄漏问题。
- tests/entrypoints/openai/completion/test_shutdown.py（模块 关闭测试；类别 test；类型 test-coverage）：微调测试超时参数，与其他 CI 环境适配，非核心变更。

关键符号：_supports_kw, supports_kw, GPUModelRunner.shutdown, EngineCore.shutdown, GPUWorker.shutdown, _wait_for_prefix_cache_reset, test_gpu_memory_rixl_hma, test_gpu_memory_no_rixl_baseline

关键源码片段

vllm/utils/func_utils.py

通过将 supports_kw 拆分为非缓存外观和缓存内部函数，在缓存键中解包 bound method，根除因 LRU 缓存绑定实例导致的内存泄漏。

```
@lru_cache
def _supports_kw(
    callable: Callable[..., object],
    kw_name: str,
    *,
    requires_kw_only: bool = False,
    allow_var_kwargs: bool = True,
) -> bool:
    """Internal cached implementation of supports_kw."""
    # 原本 supports_kw 的主体逻辑保持不变，但以底层函数为键
    params = inspect.signature(callable).parameters
    # ... 省略相同逻辑


def supports_kw(
    callable: Callable[..., object],
    kw_name: str,
    *,
    requires_kw_only: bool = False,
    allow_var_kwargs: bool = True,
) -> bool:
    """Check if a keyword is a valid kwarg for a callable."""
    # 关键修复：解包 bound method 为底层函数，避免缓存以实例为键
    # 若不解包，model.forward 等 bound method 会保持模型引用，
    # 导致 GPU 权重无法回收
    if hasattr(callable, "__func__"):
        callable = callable.__func__
    return _supports_kw(
        callable, kw_name,
        requires_kw_only=requires_kw_only,
        allow_var_kwargs=allow_var_kwargs,
    )
```

vllm/v1/worker/gpu_model_runner.py

新增 shutdown() 方法，集中释放 KV 缓存、模型权重、旋转嵌入缓存和工作空间，是 GPU 内存清理的核心。

```
def shutdown(self) -> None:
    """Release GPU tensors (model weights, KV caches, workspace) so that
    memory is reclaimable when running in the same process."""
    from vllm.model_executor.layers.rotary_embedding import _ROPE_DICT
    from vllm.v1.worker.workspace import reset_workspace_manager

    # 注意：此方法内部调用 torch.accelerator.synchronize()
    self._cleanup_profiling_kv_cache()

    # 清空编译静态前向上下文中的引用
    self.compilation_config.static_forward_context.clear()

    # 释放模型权重（赋值 None 以降低引用计数）
    self.model = None # type: ignore[assignment]

    # 清空旋转位置编码缓存
    _ROPE_DICT.clear()

    # 重置工作空间单例（释放额外分配）
    reset_workspace_manager()
```

vllm/v1/engine/core.py

在 EngineCore.shutdown() 中添加 gc.unfreeze()，与 __init__ 中的 gc.freeze() 配对，使得引擎对象在关闭后对 GC 可见。

```
def shutdown(self):
    self.structured_output_manager.clear_backend()
    if self.model_executor:
        self.model_executor.shutdown()
    if self.scheduler:
        self.scheduler.shutdown()

    # 撤销 EngineCore.__init__ 中的 gc.freeze()，
    # 使得引擎启动期间分配的对象（模型权重、KV 缓存等）
    # 重新对垃圾回收器可见。若无此行，这些对象将一直
    # 停留在 frozen 列表中，导致 in-process 模式下的 GPU
    # 内存泄漏。
    gc.unfreeze()
```

评论区精华

- SageMoore 对 * 移除的疑问：在 _supports_kw 中，原 supports_kw 的 * 参数被移除，SageMoore 询问原因。作者回复为无意疏忽，并在后续提交中还原。

- SageMoore 建议添加同步注释：在 GPUModelRunner.shutdown 调用 _cleanup_profiling_kv_cache 前，建议添加注释 # Calls torch.accelerator.synchronize()，作者已补充。
- tjtanaa 对 Dockerfile 中 fastsafetensors 重复安装的质疑：tjtanaa 指出 requirements/test/rocm.txt 已包含 fastsafetensors，Dockerfile 中再次安装可能导致版本不一致。作者承认忘记，并在后续 commit 中回退。
- orozery 对 reset_prefix_cache workaround 的设计讨论：orozery 认为当前 workaround 滥用 has_finished_requests 语义且未能完全解决竞态，建议引入 flush_transfers 连接器 API。作者尝试实现但回退到较简单的测试 workaround，orozery 认可但建议提取为独立 PR，作者选择保留。
  - 移除 * 导致关键字参数默认行为改变 (style): 作者承认是无意遗漏，已重新添加 *。
  - 添加 # Calls torch.accelerator.synchronize() 注释 (documentation): 作者已添加注释。
  - Dockerfile 中 fastsafetensors 重复安装 (other): 作者承认遗忘，并在后续 commit 中回退。
  - reset_prefix_cache workaround 的设计替代方案 (design): orozery 认可测试 workaround，但建议将其提取为独立 PR。作者选择保留在当前 PR 中。未采纳 API 方案。

风险与影响

- 风险：
  - gpu_model_runner.shutdown 依赖 _cleanup_profiling_kv_cache：该方法原本仅用于 CUDA graph profiling 清理，shutdown 复用可能在 profiling 逻辑变更时引入隐藏依赖。
  - gc.unfreeze() 调用时机：位于 EngineCore.shutdown 末尾，若前面清理未完全释放引用，部分对象可能仍保持 frozen，但视为低风险。
  - 测试workaround可能增加测试时间：dummygenerate引入额外推理，但正确性无影响。
  - UCX 版本固定：特定 commit 在未来可能需要同步上游修复。
  - in-process 模式非默认路径：多进程模式不受影响，风险范围有限。
- 影响：
  - 对 VLLM_ENABLE_V1_MULTIPROCESSING=0 用户：修复了引擎关闭后 GPU 内存泄漏，使得同一进程多次创建 / 销毁引擎成为可能，特别有利于单元测试。
  - 对 ROCm 平台 NixlConnector 用户：UCX 升级和诊断测试确保了 RIXL 内存正确释放，提高了稳定性。
  - 对系统资源管理：显式释放模型权重、KV 缓存等，减少了进程内内存浪费。
  - 影响范围：主要涉及 v1 engine 进程内关闭路径，多进程模式（默认）受影响较小。测试 workaround 仅影响特定 offloading 测试。
  - 风险标记：shutdown 依赖 profiling 清理路径，gc.unfreeze 时机依赖周边清理，测试 workaround 可能掩盖调度器问题，UCX 版本需持续跟踪兼容性，in-process 模式在 v1 中非主流路径，bound method 解包影响所有 LRU 缓存调用

关联脉络

- PR #37160 [Discussion] Fix for engine stalling in async KV offload: 在讨论中被 orozery 和作者提及，本 PR 中的测试 workaround 是基于该 PR 的讨论演化而来。
- PR #40794 [Bugfix][MoE] Unpad routed output before shared expert add: 同一作者（AndreasKaratzas）的另一修复，虽不直接相关但显示其对 ROCm 和 MoE 的关注。
- PR #40306 [ROCm][CI] Fix trust_remote_code AttributeError in EAGLE3 test: 同属 ROCm CI 修复系列，本 PR 也涉及 ROCm 平台的 CI 改进（UCX 升级、测试 workaround）。