

PR #54162 完整报告

vllm-project/vllm

[Bugfix][MRV2] Release model and KV cache on in-process engine shutdown

合并时间: 2026-08-29 13:09

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/54162>

执行摘要

- 一句话: 修复进程内引擎关闭后模型与 KV 缓存不释放
- 推荐动作: 值得精读: PR 展示了如何通过 GC 根分析定位模块级持有者, 并结合 GPU 实测数据做出设计取舍 (放弃 KV sever)。重点关注 WeakKeyDictionary 对 `get_language_model` 缓存语义的影响, 以及 finalizer 弱引用化后对编译 wrapper 清理路径的影响。与 #54246 合并前需协调重叠改动。

功能与动机

PR body 指出, 进程内模式没有进程退出兜底, “teardown has to actually drop its references”; AMD CI 在 `tests/v1/e2e/test_hybrid_chunked_prefill.py::test_qwen_mtp_mba_prefix_cache_hit_is_bounded` 中报出 `ValueError: Memory of devices devices=[0, 1, 2, 3] not free after dur_s=240.29`, 设备 0 在 240 秒内残留 24.11 GiB 且逐次轮询 bit-identical, 排除了懒回收。作者定位到两个模块级持有者: `_language_model_by_module` 强引用字典和 `LLMEngine._finalizer` 的强参数, 必须同时修复。关联 Issue #42074 描述的是同类 shutdown 泄漏 (offloader 单例), 但未覆盖这两个持有者。

实现拆解

1. 消除 Holder 1——`_language_model_by_module` 弱键化: 在 `vllm/model_executor/models/interfaces.py` 中, 将模块级缓存 `_language_model_by_module` 从 `dict[nn.Module, VllmModel]()` 改为 `weakref.WeakKeyDictionary`。该缓存以顶层模型为 key、以 `get_language_model()` 返回的子模块为 value; 子模块是 key 的 submodule, torch 模块不持有父模块反向引用, 因此 value 不会复活 key, 弱键化后 `gc.collect()` 可以真正回收模型。
2. 消除 Holder 2——`LLMEngine._finalizer` 弱引用化: 在 `vllm/v1/engine/llm_engine.py` 中, `weakref.finalize(self, LLMEngine._cleanup_instance_caches, weakref.ref(model))`, 并把 `_cleanup_instance_caches` 的签名改为接收 `weakref.ref[nn.Module]`, 先 `model_ref()` 后判空返回。原因是 CPython 的 `finalize` 会在类级注册表中强持有 args, 直到 finalizer 触发; 而 finalizer 只在 LLMEngine 被回收后才触发, 进程内模式下调用者仍持有引擎句柄, 模型因此被无限期钉住。
3. 新增两个 CPU 测试锁住语义: `tests/v1/engine/test_llm_engine_finalizer_is_weak.py` 验证 `weakref.finalize` 强持有参数的前提、清理逻辑容忍模型已死、以及模型存活时仍会遍历 `TorchCompileWithNoGuardsWrapper` 执行 `cleanup`;

tests/models/test_language_model_cache_is_weak.py 验证缓存是 WeakKeyDictionary、缓存项不钉住模型、key 被回收后条目自动消失。两者都标记 pytest.mark.cpu_test。

4. 演进与合并：最初的提交包含 MRV2 的 per-layer KV sever (free_before_shutdown) , 提交 b6b6c79 根据 GPU 实测 (sever 只降到 10.46 GiB, 弱引用修复可到 2.99 GiB) 将其回退, 只保留两个模型钉住修复; 后续两次合并 main 解决冲突。

关键文件:

- vllm/v1/engine/llm_engine.py (模块 引擎核心; 类别 source; 类型 core-logic; 符号 _cleanup_instance_caches, _get_driver_model_for_cleanup) : 核心修复: 将 finalizer 从强持有模型改为弱引用, 并让清理逻辑容忍模型已死, 是释放模型与 KV 缓存的关键。
- vllm/model_executor/models/interfaces.py (模块 模型接口; 类别 source; 类型 data-contract; 符号 _language_model_by_module) : 将模块级 _language_model_by_module 从强引用 dict 改为 WeakKeyDictionary, 消除另一处钉住顶层模型的 GC 根。
- tests/v1/engine/test_llm_engine_finalizer_is_weak.py (模块 引擎测试; 类别 test; 类型 test-coverage; 符号 _Model, test_finalize_args_are_held_strongly, Owner, test_cleanup_tolerates_a_dead_model) : 新增测试直接验证 weakref.finalize 强持有参数的前提, 以及清理逻辑对死模型和活模型的行为。
- tests/models/test_language_model_cache_is_weak.py (模块 缓存测试; 类别 test; 类型 test-coverage; 符号 _LanguageModel, embed_input_ids, _MultiModalModel, init) : 新增测试确保缓存为 WeakKeyDictionary、缓存项不钉住模型、key 被回收后项被丢弃。

关键符号: _cleanup_instance_caches, test_finalize_args_are_held_strongly, test_cleanup_tolerates_a_dead_model, test_cleanup_still_walks_a_live_model, test_cache_is_weak_keyed, test_cached_entry_does_not_pin_the_model, test_cached_entry_is_dropped_with_its_key

关键源码片段

vllm/v1/engine/llm_engine.py

核心修复: 将 finalizer 从强持有模型改为弱引用, 并让清理逻辑容忍模型已死, 是释放模型与 KV 缓存的关键。

```
# finalizer 注册: 在进程内模式 (VLLM_ENABLE_V1_MULTIPROCESSING=0) 下,
# 模型此时还可达, 但 weakref.finalize 会强持有 args, 导致模型被钉住。
# 改为传 weakref.ref(model), 让模型在引擎存活期间也可被 GC 回收。
model = self._get_driver_model_for_cleanup()
if model is not None:
    self._finalizer = weakref.finalize(
        self, LLMEngine._cleanup_instance_caches, weakref.ref(model)
    )
```

@staticmethod

```
def _cleanup_instance_caches(model_ref: "weakref.ref[nn.Module]") -> None:
    """移除钉住已编译模型的 bytecode hooks, 释放 GPU 内存。"""
```

```

from vllm.compilation.wrapper import TorchCompileWithNoGuardsWrapper

model = model_ref()
if model is None:
    # 模型已被回收，无需清理；此时其子模块 wrapper 也随之释放
    return
for module in model.modules():
    if isinstance(module, TorchCompileWithNoGuardsWrapper):
        module.cleanup()

```

vllm/model_executor/models/interfaces.py

将模块级 `_language_model_by_module` 从强引用 dict 改为 WeakKeyDictionary，消除另一处钉住顶层模型的 GC 根。

```

import weakref

# 缓存 SupportsMultiModal.get_language_model 的结果。
# 由强引用 dict 改为 WeakKeyDictionary: key 是顶层模型，value 是
# language_model 子模块；子模块不反向持有父模型，因此 value 不会复活 key。
# 这样引擎 shutdown 后模型可以被 gc.collect() 真正回收。
_language_model_by_module: "weakref.WeakKeyDictionary[nn.Module, VllmModel]" = (
    weakref.WeakKeyDictionary()
)

```

tests/v1/engine/test_llm_engine_finalizer_is_weak.py

新增测试直接验证 `weakref.finalize` 强持有参数的前提，以及清理逻辑对死模型和活模型的行为。

```

# 验证 weakref.finalize 强持有参数，这是必须改为弱引用的前提。
def test_finalize_args_are_held_strongly():
    class Owner:
        pass

    owner, model = Owner(), _Model()
    weakref.finalize(owner, lambda _m: None, model)
    ref = weakref.ref(model)

    del model
    gc.collect()
    assert ref() is not None # finalize 的强引用仍让 model 存活

    del owner
    gc.collect()
    assert ref() is None

# 清理逻辑必须容忍模型已死（弱引用解引用为 None）。
def test_cleanup_tolerates_a_dead_model():
    model = _Model()

```

```
ref = weakref.ref(model)
del model
gc.collect()
assert ref() is None
```

```
LLMEngine._cleanup_instance_caches(ref)
```

评论区精华

本 PR 没有实质 review 评论，njhill 在 issue 评论中要求 “@okorzh-amd could you rebase / resolve conflicts”，作者回复 “done” 并重新触发 CI，最终 njhill 批准。核心设计讨论都集中在 PR body：作者论证了为何放弃最初随 PR 引入的 MRV2 per-layer KV sever——#41594 的 review 结论成立，`del` 模型即可释放 KV，前提是模型真的能死；并说明与 #54246 互补而非重复，两边的修复对象不同（#54246 释放 12.0 GiB 的 KV cache，本 PR 同时释放 KV 与 8.9 GiB 权重），且都改 `tests/conftest.py` 同一处 `except Exception: pass`，需要协调合并顺序。作者还承认一个未决边界：wrapper 模块的 hook 可能独立于模型存活，但没有找到实例，权衡后不为此保留约 9 GiB VRAM。

- rebase 与冲突解决 (other): 作者完成 rebase，njhill 最终批准本 PR。
- 是否保留 MRV2 的 per-layer KV sever (design): 放弃 sever，仅保留两个模型钉住修复；与 #54246 互补，需协调 `tests/conftest.py` 的同一处修改。

风险与影响

- 风险：
 1. `LLMEngine._cleanup_instance_caches` 改为弱引用后，若模型在 finalizer 触发前已被回收，bytecode hook 清理会被跳过；但模型死亡意味着其 wrapper 子模块也随图不可达，hook 不再被引用，实际风险低。
 2. `_language_model_by_module` 从强缓存变为弱缓存，改变了 `get_language_model` 的缓存语义：若调用方在顶层模型回收后仍持有 value (`language_model` 子模块)，缓存条目会消失但值仍被外部持有，行为正确；若存在代码依赖缓存保持 key 存活，则需重新审视。
 3. 改动影响 MRV1 与 MRV2 共用路径，进程内模式下可能暴露其他隐藏的模型持有者；PR 作者也承认 wrapper 模块独立存活的理论情况未找到实例，属于已知边界。
 4. 测试均为 CPU 单测，GPU 上的回收结论依赖作者在 MI355X 上的实测；与 #54246 在 `tests/conftest.py` 的冲突若处理不当会导致重复修改或漏掉日志化 `VllmRunner.__exit__` 的改进。- 影响：进程内模式（`VLLM_ENABLE_V1_MULTIPROCESSING=0`）用户在 `engine.shutdown()` 后能真正释放 GPU 显存，修复 AMD CI 的内存检查失败；混合模型（如 Qwen3.5 MTP 路径）在 MRV1/MRV2 下的模型生命周期更符合预期，可安全反复创建 / 销毁引擎。团队协作上需要与 #54246 协调合并顺序，避免 `tests/conftest.py` 冲突；模型接口层缓存语义的变化对所有调用 `get_language_model` 的多模态 / 混合模型有潜在影响。- 风险标记：进程内引擎关闭路径变更，模型生命周期语义变化，与 #54246 的 `conftest.py` 冲突，仅 CPU 单测覆盖

关联脉络

- PR #54246 [Bugfix][MRV2] Release layer-bound KV cache memory in shutdown(): 同区域互补方案：做 per-layer KV sever；与本 PR 分别修复不同持有者，且都改 tests/conftest.py 同一处，需协调合并顺序。
- PR #41594 [MRV2] Port MRV1 KV sever into free_before_shutdown (closed): 被关闭的先前尝试；其 review 结论成立，本 PR 是其缺失的另一半（解除模型钉住）。
- PR #42074 [Bugfix] Reset offloader singleton in shutdown() to prevent GPU memory leak: 关联 Issue/PR：同类模块级持有者导致的 shutdown 内存泄漏，但不覆盖本 PR 的两个持有者。