PR #45347 完整报告

vllm-project/vllm

[BugFix] Avoid prematurely freeing cached mm encoder outputs

合并时间：2026-06-13 06:39

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/45347

执行摘要

- 一句话：修复异步调度 + 推测解码中编码器输出过早释放导致崩溃
- 推荐动作：值得精读。修复思路清晰简洁，通过减法引入"回滚余量"是优雅的处理方式。测试覆盖全面，代码注释清晰。建议关注 _free_encoder_inputs 的边界条件和 num_output_placeholders 的维护逻辑，可作为后续类似问题的设计参考。

功能与动机

Issue #38551 报告：在使用 MTP 推测解码 + 多模态输入的高并发生产场景下，引擎会因 AssertionError: Encoder cache miss 而崩溃。根本原因是调度器在 _free_encoder_inputs 中仅以 num_computed_tokens 是否超过 placeholder 末尾来决定释放，但在推测解码中 num_computed_tokens 包含未确认的 draft tokens，一旦发生回滚，已释放的缓存导致后续 gather 失败。PR 目的是在释放时考虑未确认的 placeholder 数，避免过早释放。

实现拆解

1. 修改核心释放逻辑：在 vllm/v1/core/sched/scheduler.py 的 _free_encoder_inputs 方法中，将释放条件从 start_pos + num_tokens <= request.num_computed_tokens 改为 start_pos + num_tokens <= request.num_computed_tokens - request.num_output_placeholders。这样，在推测解码场景下，即使 num_computed_tokens 已超过 placeholder 范围，只要未确认的 draft tokens 数量（num_output_placeholders）导致确认位置仍在范围内，编码器输出就会被保留。该修改仅影响是否调用 free_encoder_input，不会改变缓存管理器的其他行为。
2. 添加四个回归测试：在 tests/v1/core/test_scheduler.py 中新增四个测试用例：
   - test_free_encoder_inputs_respects_unconfirmed_placeholders：模拟推测解码下 num_output_placeholders 非零的场景，逐步推进 num_computed_tokens，验证确认位置超出范围后才释放。
   - test_free_encoder_inputs_unchanged_without_spec_decode：验证无推测解码时行为不变（num_output_placeholders 为 0，条件等价于原逻辑）。
   - test_encoder_cache_retained_across_preemption_and_resume：测试请求被抢占后恢复时，编码器缓存是否正确保留（重计算路径）。
   - test_encoder_cache_recomputed_when_evicted_during_preemption：测试编码器输入在抢占期间被逐出后，是否能在后续调度中重新计算。 测试使用 create_scheduler 和

create_requests 辅助函数，直接检查 encoder_cache_manager.get_cached_input_ids 来验证缓存状态。

关键文件：

- vllm/v1/core/sched/scheduler.py（模块 调度器；类别 source；类型 core-logic；符号 _free_encoder_inputs）：核心修复文件：修改 _free_encoder_inputs 方法中的释放条件，增加 num_output_placeholders 的折减，避免编码器输出在推测解码回滚时过早释放。
- tests/v1/core/test_scheduler.py（模块 调度器；类别 test；类型 test-coverage；符号 test_free_encoder_inputs_respects_unconfirmed_placeholders, test_free_encoder_inputs_unchanged_without_spec_decode, test_encoder_cache_retained_across_preemption_and_resume, test_encoder_cache_recomputed_when_evicted_during_preemption）：新增 4 个回归测试用例，全面覆盖推测解码回滚、非推测解码、抢占恢复等场景，确保修复正确性并防止回归。

关键符号：_free_encoder_inputs, test_free_encoder_inputs_respects_unconfirmed_placeholders, test_free_encoder_inputs_unchanged_without_spec_decode, test_encoder_cache_retained_across_preemption_and_resume, test_encoder_cache_recomputed_when_evicted_during_preemption

关键源码片段

vllm/v1/core/sched/scheduler.py

核心修复文件：修改 _free_encoder_inputs 方法中的释放条件，增加 num_output_placeholders 的折减，避免编码器输出在推测解码回滚时过早释放。

```
# vllm/v1/core/sched/scheduler.py ( 修改后 )
def _free_encoder_inputs(self, request: Request) -> None:
    cached_encoder_input_ids = self.encoder_cache_manager.get_cached_input_ids(request)
    if not cached_encoder_input_ids:
        return

    for input_id in list(cached_encoder_input_ids):
        mm_feature = request.mm_features[input_id]
        start_pos = mm_feature.mm_position.offset
        num_tokens = mm_feature.mm_position.length
        if self.is_encoder_decoder and request.num_computed_tokens > 0:
            # Whisper 场景：一旦生成第一个 token，立即释放
            self.encoder_cache_manager.free_encoder_input(request, input_id)
        elif (
            start_pos + num_tokens
            <= request.num_computed_tokens - request.num_output_placeholders
        ):
            # 核心修复：确认位置（已计算位置减去未确认的占位符数）
            # 必须超出 placeholder 范围才可释放，防止回滚后缓存缺失
            self.encoder_cache_manager.free_encoder_input(request, input_id)

# 注意：当 num_output_placeholders 为 0（非推测解码）时，
```

```
# 条件退化回原逻辑，行为保持不变。
```

tests/v1/core/test_scheduler.py

新增 4 个回归测试用例，全面覆盖推测解码回滚、非推测解码、抢占恢复等场景，确保修复正确性并防止回归。

```
# tests/v1/core/test_scheduler.py ( 新增 )
def test_free_encoder_inputs_respects_unconfirmed_placeholders():
    """Regression test for issue #38551 (rollback path)."""
    scheduler = create_scheduler(
        model="llava-hf/llava-1.5-7b-hf",
        num_speculative_tokens=3, # 启用推测解码
    )
    mm_start_pos = 50
    mm_length = 100
    mm_positions = [[PlaceholderRange(offset=mm_start_pos, length=mm_length)]]
    request = create_requests(
        num_requests=1,
        num_tokens=mm_start_pos + mm_length + 100,
        mm_positions=mm_positions,
    )[0]
    manager = scheduler.encoder_cache_manager
    manager.allocate(request, 0) # 分配编码器缓存
    mm_end = mm_start_pos + mm_length

    # 模拟一次推测步骤：生成 1 个真实 token + 3 个 draft tokens,
    # 全部未确认，因此 num_output_placeholders = 4
    request.num_output_placeholders = 4

    # 场景 1：num_computed_tokens 刚超出范围，但确认位置仍在范围内
    request.num_computed_tokens = mm_end + 1
    scheduler._free_encoder_inputs(request)
    assert manager.get_cached_input_ids(request) == {0} # 应保留

    # 场景 2：更深入，但确认位置仍在内
    request.num_computed_tokens = mm_end + 3
    scheduler._free_encoder_inputs(request)
    assert manager.get_cached_input_ids(request) == {0} # 继续保留

    # 场景 3：确认位置到达范围末尾，可以安全释放
    request.num_computed_tokens = mm_end + 4
    scheduler._free_encoder_inputs(request)
    assert manager.get_cached_input_ids(request) == set() # 释放
```

评论区精华

该 PR 无 review 评论，由 ywang96 直接批准合并。Claude 被用于生成测试代码（PR body 提及）。

- 暂无高价值评论线程

风险与影响

- 风险：风险较低。核心改动仅在一行条件判断中增加了减法项 - request.num_output_placeholders，且该字段在非推测解码场景下为 0，不影响原有行为。测试覆盖了推测解码回滚、非推测解码、抢占恢复等多种场景。需要关注的是 num_output_placeholders 的正确维护——但该字段已在推测解码流程中被其他逻辑更新，不会因本 PR 引入新问题。潜在风险：若未来有代码路径忘记更新 num_output_placeholders，可能导致释放时机不准，但此风险与 PR 本身无关。
- 影响：对用户：修复了生产环境中高并发多模态 + MTP 场景下的崩溃问题（Issue #38551），提升稳定性。对无推测解码的用户无影响（行为不变）。对系统：调度器在多模态请求生命周期内更长时间保留编码器缓存，可能略微增加缓存压力，但这是正确的 trade-off（避免崩溃比缓存压力更重要），且缓存最终会在确认位置通过后被释放。对团队：此修复明确了推测解码下 num_computed_tokens 的语义——它包含乐观推进部分，释放判断应基于确认位置。
- 风险标记：核心逻辑路径变更，需要维护 num_output_placeholders 同步

关联脉络

- PR #34220 [Core] Fix EAGLE3 encoder cache miss when disable_chunked_mm_input is set: 之前修复了一个类似的 encoder cache miss 问题（EAGLE3 场景），与当前 PR 有关联 Issue #38551 的区别中提到。
- PR #42759 [Bug] Migrate Reset cache for both v2 and v1 model runner: 涉及 v1/v2 model runner 的缓存管理重置，与编码器缓存生命周期相关。
- PR #43877 [Core][KV Connector] fix scheduler KV connector stats aggregation: 修复 scheduler 中 KV connector 统计聚合，同属 scheduler 模块的稳定性修复。