PR #48245 完整报告

vllm-project/vllm

[BugFix] Fix `num_output_placeholders` preemption underflow

合并时间：2026-07-29 21:41

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/48245

执行摘要

- 一句话：修复异步调度下 num_output_placeholders 下溢崩溃
- 推荐动作：该 PR 值得精读，尤其是 PipelinedEngine 测试模拟器的设计，展示了如何在不依赖 GPU 的情况下验证 PP 下的抢占竞争条件。修复方案中"保留 stale output 但跳过计数器更新"的权衡体现了对 spec decode 正确性的深入理解。

功能与动机

Fix an EngineCore crash under async scheduling + KV-pressure preemption (most visible with spec decode / PP at high concurrency). Replaces PR #47900 whose discard-based approach caused CI failures. Also potentially fixes issue #41190.

实现拆解

1. 新增状态字段（vllm/v1/request.py）：用 num_stale_output_tokens 和 drop_stale_output 替换旧的 async_tokens_to_discard。
2. 修改 _preempt_request（vllm/v1/core/sched/scheduler.py）：新增 drop_stale_output 参数，将 num_in_flight_tokens 转移为 num_stale_output_tokens，清零 num_output_placeholders。
3. 调度时跳过未消耗完 stale 的请求（scheduler.py 中 schedule 方法）：若 num_stale_output_tokens > 0 且不丢弃，则延缓重新调度。
4. 修改 _update_request_with_output（vllm/v1/core/sched/async_scheduler.py）：新增 is_stale 参数，stale 输出跳过 placeholder 更新，移除旧的 async_tokens_to_discard。
5. 配套测试：引入 PipelinedEngine 模拟 PP 调度窗口，新增两个回归测试验证无下溢且 acceptance 不变。

关键文件：

- vllm/v1/core/sched/scheduler.py（模块 调度器；类别 source；类型 core-logic；符号 _preempt_request, update_from_output）：核心抢占逻辑修改：_preempt_request 标记 stale output，update_from_output 同步减少 stale 计数，schedule 跳过未消耗完 stale 的请求。
- vllm/v1/core/sched/async_scheduler.py（模块 异步调度器；类别 source；类型 core-logic；符号 _update_request_with_output）：修改 _update_request_with_output 以处理 stale output，避免下溢断言。移除旧的 async_tokens_to_discard 逻辑。

- vllm/v1/request.py（模块 请求模型；类别 source；类型 core-logic；符号 init）：在 Request 类中替换 async_tokens_to_discard 为 num_stale_output_tokens 和 drop_stale_output。
- tests/v1/core/test_async_scheduler.py（模块 异步调度测试；类别 test；类型 test-coverage；符号 PipelinedEngine, init, _schedule, _process_oldest_step）：新增 PipelinedEngine 模拟器和两个回归测试，验证无下溢且 acceptance 不变。
- tests/v1/core/test_scheduler.py（模块 调度测试；类别 test；类型 test-coverage；符号 test_preemption_re_records_prefix_cache_query）：调整测试以适配新的 output 处理流程，确保 preemption 后 stale output 正确消耗。

关键符号：_preempt_request, update_from_output, _update_request_with_output, Request.init

关键源码片段

vllm/v1/core/sched/scheduler.py

核心抢占逻辑修改：_preempt_request 标记 stale output，update_from_output 同步减少 stale 计数，schedule 跳过未消耗完 stale 的请求。

```
def _preempt_request(
    self, request: Request, timestamp: float, drop_stale_output: bool = False
) -> None:
    '''Preempt a request, marking in-flight outputs as stale.'''
    assert request.status == RequestStatus.RUNNING
    self._free_request_blocks(request)
    self.encoder_cache_manager.free(request)
    self._inflight_prefills.discard(request)
    request.status = RequestStatus.PREEMPTED
    request.num_computed_tokens = 0
    if request.spec_token_ids:
        request.spec_token_ids = []

    # Transfer all in-flight tokens to stale counter.
    # Use assignment (not accumulate) because `num_in_flight_tokens`
    # already includes any previous stale share.
    request.drop_stale_output = drop_stale_output or (
        request.drop_stale_output and request.num_stale_output_tokens > 0
    )
    request.num_stale_output_tokens = request.num_in_flight_tokens
    request.num_output_placeholders = 0
    request.num_preemptions += 1
    if self.log_stats:
        request.record_event(EngineCoreEventType.PREEMPTED, timestamp)
```

vllm/v1/core/sched/async_scheduler.py

修改 _update_request_with_output 以处理 stale output，避免下溢断言。移除旧的 async_tokens_to_discard 逻辑。

```
def _update_request_with_output(
    self, request: Request, new_token_ids: list[int], is_stale: bool = False
) -> tuple[list[int], bool]:
    '''
    Updates request state with new token ids.
    If `is_stale`, skip placeholder decrement to avoid underflow.
    '''
    status_before_update = request.status
    new_token_ids, stopped = super()._update_request_with_output(
        request, new_token_ids
    )

    # Placeholders were zeroed by `_preempt_request`;
    # stale delivery must not decrement them.
    if not is_stale:
        request.num_output_placeholders -= len(new_token_ids)
        assert request.num_output_placeholders >= 0

    # Cache tokens only if the request was still RUNNING at output start.
    if status_before_update == RequestStatus.RUNNING:
        self.kv_cache_manager.cache_blocks(
            request,
            request.num_computed_tokens - request.num_output_placeholders,
        )
    return new_token_ids, stopped
```

评论区精华

ivanium 提出 Codex 提示的边缘情况：若 stale output 使可恢复流在 PREEMPTED 状态下停止，_handle_stopped_request() 可能将 session 重新入队到 skipped_waiting，但后续清理会将其从两个队列中移除，导致 session 悬空。ivanium 随后表示已通过 Codex 修复该问题，PR 最终获批。

- Stale output 导致 session 悬空的风险 (design): ivanium 随后表示已让 Codex 修复该边缘情况，最终 PR 获得批准。

风险与影响

- 风险：改动集中在抢占路径，不影响正常调度。新状态 num_stale_output_tokens/drop_stale_output 需与 num_in_flight_tokens 协同维护，若不一致可能导致某些输出被错误处理。update_from_output 中 output_is_stale 判断逻辑保持对称是维护关键。经过新增的 PP 模拟测试和 e2e 测试，回归风险低。
- 影响：对用户透明，仅修复崩溃。影响场景为 v1 async scheduling 下触发 KV 抢占的请求，尤其是使用 speculative decoding 或 pipeline parallelism 时。测试覆盖了这些场景，无行为变更。
- 风险标记：抢占路径逻辑复杂，新增状态变量需谨慎维护

关联脉络

- PR #47900 [BugFix] Discard in-flight async output frames on preemption: 同一 bug 的前次修复尝试，方案不同导致 CI 失败，本 PR 取代之。
- PR #46066 [BugFix] Handle reset_prefix_cache with async scheduling: 类似场景的修复，但仅处理了 reset_prefix_cache 路径，本 PR 统一处理 KV-pressure 抢占。