

PR #46974 完整报告

vllm-project/vllm

[BugFix][MRV2] Ensure all req slots are accounted for when scheduling

合并时间: 2026-07-03 01:19

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/46974>

执行摘要

- 一句话: 修复 V2 ModelRunner 调度时请求槽计数遗漏
- 推荐动作: 建议所有使用 V2 Model Runner 的团队同步此修复。测试文件的 WorkerSlots 模拟类设计值得学习, 可复用至其他需要验证调度器余额的场景。PR 提交清晰, 改动集中, 可快速 Code Review。

功能与动机

V2 Model Runner 使用固定大小的槽位数组 (大小 = max_num_seqs), 从请求出现在 scheduled_new_reqs 到出现在 finished_req_ids/preempted_req_ids 期间占用一个槽。调度器必须保证: 任何仍持有一个工作槽的请求都计入准入限制。PR 指出两种边界情况违反了该不变式: 等待流式输入的请求 (WAITING_FOR_STREAMING_REQ) 不在运行队列但占用槽; 被外部抢占的请求 (如 reset_prefix_cache) 可能被新请求覆盖。修复并添加回归测试。

实现拆解

实现步骤

1. 新增状态跟踪: 在 Scheduler.__init__ (vllm/v1/core/sched/scheduler.py) 中初始化 self.reset_preempted_req_ids: set[str], 用于累积两次 schedule() 调用之间发生的抢占请求 ID。在 _preempt_request 方法中, 将请求 ID 加入该集合。
2. 修正准入计数: 在 schedule() 主循环中, 将运行队列长度判断改为 len(self.running) + self.num_waiting_for_streaming_input >= self.max_num_running_reqs, 将等待流式输入的请求也视为占用槽位, 从而防止在它们仍在占用时额外准入新请求。
3. 传递抢占 ID: 在构建 SchedulerOutput 时, 将 preempted_req_ids 参数改为 self.reset_preempted_req_ids, 确保之前步骤中发生的抢占也被包含, 而非仅当前步骤抢占了哪些。
4. 正确清空: 在 _update_after_schedule 中, 除了清空 finished_req_ids, 也清空 self.reset_preempted_req_ids, 保证状态在步骤间重置 (通过创建新集合避免引用问题)。
5. 添加回归测试: 新增 tests/v1/core/test_worker_slot_overflow.py, 包含 WorkerSlots 类, 忠实模拟工作槽行为 (apply 方法在溢出时抛出相同断言错误)。三个测试用例分别覆盖流式暂停不超限、reset_prefix_cache 优先级模式不超限、同一步骤抢占后恢复并重新添加的场景。

关键文件:

- tests/v1/core/test_worker_slot_overflow.py (模块 槽位测试; 类别 test; 类型 test-coverage; 符号 WorkerSlots, init, apply, _model_runner_output) : 新增的回归测试文件, 覆盖了导致“No free indices”断言失败的两个边界场景, 并通过 WorkerSlots 模拟类验证调度器槽位计数的正确性。
- vllm/v1/core/sched/scheduler.py (模块 调度器; 类别 source; 类型 core-logic) : 包含核心调度器变更: 新增 reset_preempted_req_ids 跟踪外部抢占; 修正准入计数包含等待流式输入请求; 以及在 SchedulerOutput 中正确传递抢占 ID。

关键符号: WorkerSlots.init, WorkerSlots.apply, _model_runner_output, test_streaming_pause_does_not_over_admit_worker_slot, test_reset_prefix_cache_priority_does_not_over_admit_worker_slot, test_reset_prefix_cache_same_step_resume_purges_then_re_adds, Scheduler.init, Scheduler.schedule, Scheduler._preempt_request, Scheduler._update_after_schedule

关键源码片段

tests/v1/core/test_worker_slot_overflow.py

新增的回归测试文件, 覆盖了导致“No free indices”断言失败的两个边界场景, 并通过 WorkerSlots 模拟类验证调度器槽位计数的正确性。

```
class WorkerSlots:
    """Faithful model of the V2 model runner's request-slot accounting.

    Mirrors ``GPUModelRunner.finish_requests`` + ``add_requests`` and
    ``RequestState.add_request``/``remove_request`` (the ``free_indices`` pool).
    ``apply`` raises ``AssertionError("No free indices")`` exactly where the real
    worker would, so a scheduler over-admission surfaces as the same failure.
    """

    def __init__(self, max_num_reqs: int):
        self.max_num_reqs = max_num_reqs
        self.occupied: set[str] = set()

    def apply(self, scheduler_output) -> None:
        # finish_requests: free finished and preempted slots first.
        freed = set(scheduler_output.finished_req_ids)
        freed |= set(scheduler_output.preempted_req_ids)
        self.occupied -= freed

        # update_requests: cached reqs must already own a slot, else the
        # worker's req_id_to_index lookup would KeyError.
        for req_id in scheduler_output.scheduled_cached_reqs.req_ids:
            assert req_id in self.occupied, f"cached req {req_id!r} has no worker slot"

        # add_requests: new (and, under V2, resumed) reqs claim a slot. The
        # same-id _remove_request guard frees a stale slot for the same id.
        for new_req in scheduler_output.scheduled_new_reqs:
            self.occupied.discard(new_req.req_id)
```

```

        assert len(self.occupied) < self.max_num_reqs, "No free indices"
        self.occupied.add(new_req.req_id)

def _model_runner_output(req_ids: list[str], sampled: list[list[int]]):
    return ModelRunnerOutput(
        req_ids=req_ids,
        req_id_to_index={req_id: i for i, req_id in enumerate(req_ids)},
        sampled_token_ids=sampled,
        logprobs=None,
        prompt_logprobs_dict={},
        pooler_output=[],
    )

```

vllm/v1/core/sched/scheduler.py

包含核心调度器变更：新增 `reset_preempted_req_ids` 跟踪外部抢占；修正准入计数包含等待流式输入请求；以及在 `SchedulerOutput` 中正确传递抢占 ID。

```

# In __init__:
self.reset_preempted_req_ids: set[str] = set()

# In schedule():
# Paused streaming sessions (WAITING_FOR_STREAMING_REQ) are not
# in `running` but still hold a model-runner request slot.
num_running = len(self.running) + self.num_waiting_for_streaming_input
if num_running >= self.max_num_running_reqs:
    break

# In SchedulerOutput construction:
preempted_req_ids=self.reset_preempted_req_ids,

# In _preempt_request:
self.reset_preempted_req_ids.add(request.request_id)

# In _update_after_schedule:
self.finished_req_ids = set()
self.reset_preempted_req_ids = set()

```

评论区精华

该 PR 无 review 评论线程。审定者 WoosukKwon 直接批准合并。PR body 由作者阐述清楚动机和方案。

- 暂无高价值评论线程

风险与影响

- 风险：风险较低。主要变更是对调度器计数逻辑的修正，并且新增了专门测试覆盖。可能的风险在于：若 `num_waiting_for_streaming_input` 计数不准确（例如在其他地方未同步更新）

，仍可能导致误判；reset_preempted_req_ids 可能与其他抢占逻辑（如正常抢占）重叠，导致重复报告，但测试表明其行为正确。另外，变更仅影响 V2 Model Runner 路径，默认未对所有模型启用。

- 影响：影响使用 V2 Model Runner（默认未对所有模型启用）的用户，修复了在流式暂停和强制清空缓存时可能发生的断言失败，提升了系统稳定性。对系统性能无负面影响。对团队而言，新增测试可作为未来调度器变更的回归保护。
- 风险标记：核心调度逻辑修改，边界场景修复，新增测试覆盖

关联脉络

- PR #44443 [ModelRunner V2] Enable by default for all dense models: MRV2 默认启用后更容易触发本 PR 修复的槽溢出边界问题，两者共同推进 MRV2 的稳定性。
- PR #47428 [ModelRunner V2] Fix Mamba2 crash on non-spec-decode: 同为 V2 ModelRunner 的 bugfix，表明 MRV2 功能正在持续完善。