

PR #53962 完整报告

vllm-project/vllm

[Bugfix][Scheduler] Don't pad spec decode up to `max\_model\_len`

合并时间: 2026-08-27 23:30

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/53962>

执行摘要

- 一句话: 修复 spec decode 边界 padding 导致 max_model_len 溢出崩溃
- 推荐动作: 值得精读。该 PR 虽小, 但揭示了 v1 调度器中 waiting loop 与 running loop 对 max_model_len 上限语义不一致的根因, 以及“宁可跳过 padding、不可跳过请求”的调度取舍; 对理解 async scheduling 与 spec decode 的交互、以及如何编写边界回归测试都有参考价值。建议关注 aaron-seq 提出的测试覆盖扩展建议 (同步路径、num_spec=3) 是否在后续 PR 中落地。

功能与动机

PR body 明确指出: 等待循环中应用到单 token decode 上的 uniform spec-decode padding 允许 num_computed_tokens 恰好达到 max_model_len; 而运行循环的 max_model_len 上限会减去本步将采样的 token 数, 于是该值变为负数; 在异步调度下, 请求会在本应终止它的输出之前再次被调度, num_scheduled_tokens 为负数导致模型 runner 在 np.repeat 中崩溃。若不修复而直接 break 跳过该请求, 请求将永远无法推进从而饿死, 并 head-of-line 阻塞队列, 因此需要改为不 padding 继续调度。

实现拆解

1. 变更入口: vllm/v1/core/sched/scheduler.py 的 Scheduler.schedule() 等待循环 (waiting loop) 中, 原先无条件将单 token decode 的 num_new_tokens 置为 1 + self.num_spec_tokens, 仅检查 num_computed_tokens + num_new_tokens > self.max_model_len; 由于运行循环会再减去 num_sampled_tokens_per_step, 恰好铺满 max_model_len 时下一轮运行上限为负。
2. 核心逻辑改造: 引入局部变量 padded_num_tokens = 1 + self.num_spec_tokens, 将 padding 判定改为正向条件——仅当 num_computed_tokens + padded_num_tokens + self.num_sampled_tokens_per_step <= self.max_model_len 时才进入 padding 分支; request_token_budget 检查移入分支内部, 预算不足时仍 break (宁可不调度也不做未 padding 的调度), 而容量不足时则不再 break, 让 num_new_tokens 保持为 1, 以真实 token 继续调度。这样既优先保留 cudagraph 完整性, 又避免容量边界上请求被饿死和 head-of-line 阻塞。
3. 测试配套: tests/v1/core/test_async_scheduler.py 新增 test_no_spec_decode_padding_up_to_max_model_len, 模拟 disaggregated decode worker 场景 (请求除最后一个 prompt token 外全部已计算), 验证边界请求以 1 个

token 参与调度、不进入 `scheduled_spec_decode_tokens`、最终以 `FINISHED_LENGTH_CAPPED` 正常结束；`tests/v1/core/utils.py` 的 `create_scheduler` 将 `max_model_len` 同步传入 `ModelConfig`，因为调度器实际读取的是 `model_config.max_model_len` 而非 `SchedulerConfig` 中的值，避免测试配置不一致。

关键文件：

- `vllm/v1/core/sched/scheduler.py`（模块 调度器；类别 source；类型 core-logic；符号 `Scheduler.schedule`）：核心修复文件：更改等待循环中 spec-decode padding 的判定逻辑，预留采样 token 容量，容量不足时不再 break 而是以真实 token 继续调度。
- `tests/v1/core/test_async_scheduler.py`（模块 调度测试；类别 test；类型 test-coverage；符号 `test_no_spec_decode_padding_up_to_max_model_len`）：新增针对性回归测试，模拟 disaggregated decode worker 边界场景，锁定容量恰好不足时不再 padding、请求正常结束的行为。
- `tests/v1/core/utils.py`（模块 测试工具；类别 test；类型 test-coverage；符号 `create_scheduler`）：测试工具配套修正：将 `create_scheduler` 的 `max_model_len` 传播给 `ModelConfig`，因为调度器读取的是 `model_config.max_model_len`。

关键符号：`Scheduler.schedule`, `test_no_spec_decode_padding_up_to_max_model_len`, `create_scheduler`

关键源码片段

`tests/v1/core/test_async_scheduler.py`

新增针对性回归测试，模拟 disaggregated decode worker 边界场景，锁定容量恰好不足时不再 padding、请求正常结束的行为。

```
# tests/v1/core/test_async_scheduler.py —— 新增边界回归测试
```

```
def test_no_spec_decode_padding_up_to_max_model_len():  
    """Uniform spec-decode padding 必须为采样 token 预留容量。
```

```
  
    把单 token decode padding 到恰好 max_model_len 会使下一轮  
    running-loop 上限为负（异步调度在终止输出前再次调度该请求），  
    负的 num_scheduled_tokens 会让模型 runner 崩溃；此处验证  
    请求被以未 padding 形式调度、并正常由长度上限终止。  
    """
```

```
    max_model_len = 32  
    num_spec = 1  
    scheduler = create_scheduler(  
        async_scheduling=True,  
        max_model_len=max_model_len,  
        num_speculative_tokens=num_spec,  
        speculative_method="ngram_gpu",  
    )
```

```
  
    # 先安排一个正在运行的 decode，使 padding 的  
    # “已有调度请求且本步无 prefill”前置条件成立。
```

```

(filler,) = create_requests(num_requests=1, num_tokens=4, req_ids=["filler"])
scheduler.add_request(filler)
sched_output = scheduler.schedule()
scheduler.update_from_output(sched_output, _make_model_runner_output(sched_output))

# 边界请求: 除最后一个 prompt token 外全部已计算,
# 模拟 disaggregated decode worker 的典型到达形态。
(request,) = create_requests(
    num_requests=1, num_tokens=max_model_len - num_spec, req_ids=["boundary"]
)
request.num_computed_tokens = request.num_tokens - 1
scheduler.add_request(request)

sched_output = scheduler.schedule()
# 只能调度 1 个真实 token, 且不能进入 spec decode padding。
assert sched_output.num_scheduled_tokens["boundary"] == 1
assert "boundary" not in sched_output.scheduled_spec_decode_tokens
assert request.num_computed_tokens < max_model_len

scheduler.update_from_output(sched_output, _make_model_runner_output(sched_output))
assert request.status == RequestStatus.FINISHED_LENGTH_CAPPED

```

评论区精华

review 中 aaron-seq 对修复方向给出了正面验证: 预留 `num_sampled_tokens_per_step` 是与运行路径 cap 对齐的正确修复, 正向条件比原来的双重否定更易读; 他还确认了 fall through 安全性——当 padding 被跳过时, `request_token_budget` 至少为 1 (等待循环已保证 `token_budget > 0` 且 `input_budget > draft_slots`), 因此后续 `assert num_new_tokens > 0` 不会触发。三个 inline 评论均为非阻塞建议: 一是建议在注释中说明 running 路径已减去 `num_sampled_tokens_per_step`、两条路径因此一致, 防止日后被无意回退; 二是指出 `utils.py` 的新注释在 `max_model_len=None` 时并不成立 (ModelConfig 从模型派生 2048 而 SchedulerConfig 取 `max_num_batched_tokens`), 应缩窄注释而非改代码; 三是建议融合 PR#53812 中 parametrize 1 和 3 的边界测试, 同时锁定同步路径 `_bookkeeping_sync` 断言与异步路径两种症状。

- waiting loop 预留 `num_sampled_tokens_per_step` 的正确性与注释建议 (correctness): 修复本身获认可; 注释建议作为非阻塞意见提出, 作者未在合入前修改。
- `utils.py` 中 `max_model_len=None` 时注释与实际行为不符 (question): 评论者认为应缩小注释范围, 代码保持现状; 无后续修改记录。
- 建议融合 PR#53812 的 parametrize 边界测试覆盖同步路径与 `num_spec=3` (testing): 作为非阻塞建议提出, 当前 PR 未包含该扩展。

风险与影响

- 风险: 行为变更: 容量不足的 step 不再被跳过, 而是以未 padding 的真实 decode 继续调度, 会破坏该 step 的 spec decode cudagraph 对齐, 带来轻微性能回退, 但仅发生在请求接近 `max_model_len` 的边界场景。回归风险: 修复依赖

num_sampled_tokens_per_step 与 num_spec_tokens 的语义关系，若未来引入多采样 token 或动态 spec decode 配置变化，需要重新验证；新增测试只覆盖 num_spec=1 的异步路径，同步路径与 num_spec=3 的边界未被覆盖。测试工具一致性：utils.py 的注释在 max_model_len=None 时与实际行为不符（ModelConfig 从模型派生 2048、SchedulerConfig 取 max_num_batched_tokens），存在误导但不会影响显式传参的测试。兼容性：不影响默认非 spec decode 路径，影响面限定在 spec decode + asyncscheduling + 长度边界请求。

- 影响：对用户：修复了使用 speculative decoding + v1 异步调度时、请求长度接近 max_model_len 场景下的模型 runner 崩溃（np.repeat 报错），尤其是 KV offload、disaggregated prefill 等所有 token 几乎全部预计算的请求；此类请求从崩溃变为正常以 FINISHED_LENGTH_CAPPED 结束。对系统：等待循环在容量不足时不再 break，杜绝了边界请求饿死与 head-of-line 阻塞，提升了调度鲁棒性。对团队：该 PR 取代了此前未合并的 #50342 与 #53812，统一了修复口径和测试策略，降低了重复 PR 的维护成本。
- 风险标记：核心调度路径变更，边界容量场景，测试仅覆盖 num_spec=1，注释与实际行为存在偏差

关联脉络

- PR #50342 Similar spec decode padding fix PR: PR body 明确说明本 PR 取代 #50342，是同一问题的前次修复尝试。
- PR #53812 Similar spec decode padding fix PR: PR body 明确说明本 PR 取代 #53812；review 中建议借鉴其 parametrize 1 和 3 的边界测试。
- PR #50591 Related scheduler PR mentioned in issue comment: 评论者 sungsooha 在 PR 评论中请作者顺带查看 #50591，可能涉及同一调度区域的相邻问题。