PR #30573 完整报告

sgl-project/sglang

Configurable decode retraction order

合并时间：2026-07-10 08:55

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/30573

执行摘要

- 一句话：新增可配置的 *decode* 请求回收策略
- 推荐动作：值得精读，尤其是 _get_decode_retraction_order 的设计和 priority 策略中的优先级方向处理。该 PR 为调度器引入了一种可扩展的回收策略机制，未来可以类似地添加更多策略。

功能与动机

当 KV *cache* 在 *decode* 阶段填满时，请求被回收以释放内存。原有策略总是优先回收短输出、长输入的请求，这并非总是符合期望——某些部署希望优先回收低优先级的工作负载。

实现拆解

1. 在 server_args.py 中添加配置项：定义 RETRACTION_POLICY_CHOICES = ["length", "priority"]，新增 retraction_policy 字段，默认值为 "length"，并加入校验逻辑：当 retraction_policy == "priority" 时必须同时启用 *--enable-priority-scheduling*。
2. 在 schedule_batch.py 中提取排序逻辑：将 retract_decode 方法中原有的内联排序提取为 _get_decode_retraction_order 静态方法。该方法接收 reqs、server_args 和 allow_policy_sort 参数，返回从最想保留到最想丢弃的索引列表。当 allow_policy_sort=False（如 *spec decode* 场景）时直接返回未排序的索引列表。
3. 实现两种排序策略：
   - length 策略：使用 length_key 函数，按 (len(output_ids), -len(origin_input_ids)) 排序，保留原有行为。
   - priority 策略：使用 retraction_key 函数，先按优先级（配合 --schedule-low-priority-values-first 方向），再以 length_key 作为次级排序键。
4. 更新 retract_decode 方法：调用 _get_decode_retraction_order 代替原有的内联排序逻辑。
5. 添加单元测试：虽然后续提交移除了测试文件，但 PR 初始包含了对两种策略、优先级方向、长度 tie breaker 以及 *spec decode* 路径的测试覆盖。

关键文件：

- python/sglang/srt/managers/schedule_batch.py（模块 调度器；类别 source；类型 *core-logic*；符号 _get_decode_retraction_order, length_key, retraction_key）：包含核

心的 _get_decode_retraction_order 静态方法实现，定义了 length_key 和retraction_key 函数，并修改了 retract_decode 方法以使用新的排序逻辑。

- python/sglang/srt/server_args.py（模块 配置；类别 source；类型 configuration）：新增 retraction_policy 配置项及其校验逻辑，定义了 RETRACTION_POLICY_CHOICES 常量。

关键符号：_get_decode_retraction_order, length_key, retraction_key, retract_decode

关键源码片段

python/sglang/srt/managers/schedule_batch.py

包含核心的 _get_decode_retraction_order 静态方法实现，定义了 length_key 和 retraction_key 函数，并修改了 retract_decode 方法以使用新的排序逻辑。

```
# python/sglang/srt/managers/schedule_batch.py
# 新增的静态方法，用于确定 decode 阶段回收请求的顺序
# 返回的列表从最想保留到最想丢弃排序，retract_decode 从末尾开始 pop
@staticmethod
def _get_decode_retraction_order(
    reqs: List[Req], server_args: ServerArgs, *, allow_policy_sort: bool
) -> List[int]:
    sorted_indices = list(range(len(reqs)))

    # 当 allow_policy_sort 为 False 时（如 spec decode 场景），
    # 只能从尾部回收，直接返回原始顺序
    if not allow_policy_sort:
        return sorted_indices

    # 定义 length 策略的 key：优先回收 output_ids 长度短、origin_input_ids 长度长的请求
    def length_key(req: Req) -> Tuple[int, int]:
        return (len(req.output_ids), -len(req.origin_input_ids))

    if server_args.retraction_policy == "priority":
        # priority_sign 用于对齐 --schedule-low- 优先级 - 值 -first 的方向
        priority_sign = 1 if server_args.schedule_low_priority_values_first else -1

        def retraction_key(req: Req) -> Tuple[int, int, int]:
            priority = req.priority
            if priority is None:
                # 优先级为 None 时，根据方向选择最大或最小整数作为默认值
                priority = (
                    sys.maxsize
                    if server_args.schedule_low_priority_values_first
                    else -sys.maxsize - 1
                )
            # 使用 -priority_sign * priority 确保方向正确，然后以 length_key 作为次级排序
            return (priority * (-priority_sign), *length_key(req))

        sorted_indices.sort(
            key=lambda i: retraction_key(reqs[i]),
```

```
            reverse=True,
        )
        return sorted_indices

    # 默认的 length 策略
    sorted_indices.sort(
        key=lambda i: length_key(reqs[i]),
        reverse=True,
    )
    return sorted_indices
```

python/sglang/srt/server_args.py

新增 retraction_policy 配置项及其校验逻辑，定义了 RETRACTION_POLICY_CHOICES 常量。

```
# python/sglang/srt/server_args.py
# 新增的配置选项列表
RETRACTION_POLICY_CHOICES = ["length", "priority"]

# 在 ServerArgs 类中新增字段
retraction_policy: A[
    str,
    Arg(
        help=(
            "The decode retraction policy to use when the KV cache is full. "
            "'length' preserves the existing behavior and retracts short-output, "
            "long-input requests first. 'priority' retracts lower-priority "
            "requests first, using the same priority direction as priority "
            "scheduling."
        ),
        choices=RETRACTION_POLICY_CHOICES,
    ),
] = "length"

# 在 check_server_args 方法中增加校验
if self.retraction_policy == "priority" and not self.enable_priority_scheduling:
    raise ValueError(
        "--retraction-policy priority requires --enable-priority-scheduling"
    )
```

评论区精华

无实质 review 讨论。作者 self-approved。

- 暂无高价值评论线程

风险与影响

- 风险：

1. 回归风险：length 策略下的排序逻辑保留了原有 (len(output_ids), -len(origin_input_ids)) 的排序顺序，且仅在 allow_policy_sort=True 时应用，不影响 spec decode 路径，回归风险较低。
2. 配置依赖风险：priority 策略依赖于 --enable-priority-scheduling，若未同时启用会抛出 ValueError，属于安全设计。
3. 优先级处理：当请求的 priority 为 None 时，代码根据 schedule_low_priority_values_first 设置了 sys.maxsize 或 -sys.maxsize - 1 作为默认值，确保行为可预测。 - 影响：对用户的影响：新增的 --retraction-policy 参数允许用户根据部署需求选择回收策略。启用 priority 策略同时需要开启 --enable-priority-scheduling，使优先级调度和回收策略保持一致。对系统的影响：仅修改了调度器中回收阶段的排序逻辑，不影响其他模块。对团队的影响：提升了调度器的灵活性，便于满足不同 SLA 需求。 - 风险标记：暂无

关联脉络

- PR #29408 Avoid implicit field-based side channel in Scheduler planning: 同样涉及调度器决策逻辑的改进，体现了调度器模块持续优化。
- PR #30460 [DeepSeek V2] Reorder dual-stream MoE to main-first to avoid CUDA graph stream explosion: 同样涉及调度器行为调整，关注 decode 阶段性能。
- PR #30339 [AMD] Fix stale SWA ring buffer on radix prefix reuse for DeepSeek-V4 with unified_kv backend: 涉及 decode 阶段的调度策略修复，与 retraction policy 同属调度器内存管理范畴。