PR #25980 完整报告

sgl-project/sglang

Fix spec v2 stop output boundary

合并时间：2026-06-09 15:32

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25980

执行摘要

- 一句话：修复 spec v2 停止边界，确保多 token 接受时输出正确截断
- 推荐动作：值得精读。展示了如何在推测解码中精确处理 stop 边界，以及如何通过精细化测试覆盖多种边界情况。特别关注 _locate_str_stop_finished_len 的扫描算法和 trim_matched_stop 中 no_stop_trim 的处理。

功能与动机

This PR fixes a stop boundary issue under spec v2. When spec v2 accepts multiple tokens in one step, a stop string may appear before the end of the accepted tokens. In that case, tokens after the matched stop should not be emitted. This is an enhancement to #23802: after the stop match is detected, this PR ensures the final emitted output is cut at the actual stop boundary.

实现拆解

1. 抽取 tail_len 计算：在 schedule_batch.py 中新增 _stop_match_tail_len 方法，将 tail_len 计算逻辑从 tail_str 中提取出来，供后续 _locate_str_stop_finished_len 复用。
2. 新增精确定位方法：新增 _locate_str_stop_finished_len 方法，在新接受 token 范围内逐步 decode 并检查 stop 字符串 /regex，返回 stop 在 output_ids 中的精确结束位置。通过从 len(token_window) - new_accepted_len + 1 开始循环，节省 decode 开销。
3. 修改 finish 检查：在 _check_str_based_finish 中，当 stop 字符串在 tail_str 中匹配时，调用 _locate_str_stop_finished_len 设置 finished_len，确保后续 output_ids_through_stop 只包含 stop 之前的 token。
4. 完善 detokenizer 逻辑：修改 detokenizer_manager.py 的 trim_matched_stop 方法，正确处理 no_stop_trim 参数：当 no_stop_trim=True 时保留 stop 字符串但截断后续内容，当 no_stop_trim=False 时移除 stop 字符串及其后内容。
5. 测试覆盖：新增 test_trim_matched_stop.py 测试 detokenizer 行为，增强 test_stop_str_speculative.py 测试多种 stop 边界场景（midchunk、chunk end、跨 token、regex 等），均注册为 CPU 单元测试。

关键文件：

- python/sglang/srt/managers/schedule_batch.py（模块 请求调度；类别 source；类型 core-logic；符号 _stop_match_tail_len, _locate_str_stop_finished_len, matched）：核

心变更：新增 _stop_match_tail_len 和 _locate_str_stop_finished_len 方法，修改 _check_str_based_finish 以设置 finished_len，是修复输出边界的主要逻辑。

- python/sglang/srt/managers/detokenizer_manager.py（模块 输出处理；类别 source；类型 core-logic；符号 trim_matched_stop）：修改 trim_matched_stop 方法，正确处理 no_stop_trim 参数，是输出截断的配套变更。
- test/registered/unit/managers/test_trim_matched_stop.py（模块 测试；类别 test；类型 test-coverage；符号 _trim, TestTrimMatchedStop, test_no_finished_reason_returns_output, test_no_matched_returns_output）：新增测试文件，全面覆盖 trim_matched_stop 在 spec 场景下的各种边界情况。
- test/registered/unit/managers/test_stop_str_speculative.py（模块 测试；类别 test；类型 test-coverage；符号 _make_req, test_stop_str_midchunk_finishes, test_no_stop_does_not_finish, test_stop_str_midchunk）：增强测试，增加对 _locate_str_stop_finished_len 多种分支的测试（midchunk、chunk end、非 spec、跨 token、regex）。

关键符号：_stop_match_tail_len, _locate_str_stop_finished_len, _check_str_based_finish, trim_matched_stop

关键源码片段

python/sglang/srt/managers/schedule_batch.py

核心变更：新增 _stop_match_tail_len 和 _locate_str_stop_finished_len 方法，修改 _check_str_based_finish 以设置 finished_len，是修复输出边界的主要逻辑。

```
def _locate_str_stop_finished_len(
    self,
    new_accepted_len: int,
    *,
    stop_str: Optional[str] = None,
    stop_regex: Optional[str] = None,
) -> int:
    """将匹配的 stop 字符串/regex 映射到 output_ids 长度（包含 stop）"""
    def matched(text: str) -> bool:
        if stop_str is not None:
            return stop_str in text
        return re.search(stop_regex, text) is not None

    tail_len = self._stop_match_tail_len(new_accepted_len)
    start = len(self.output_ids) - tail_len
    token_window = self.output_ids[start:]

    # 只检查包含至少一个新接受 token 的前缀，因为旧前缀在前一步已检查过
    for token_count in range(
        max(1, len(token_window) - new_accepted_len + 1), len(token_window)
    ):
        if matched(self.tokenizer.decode(token_window[:token_count])):
            return start + token_count
```

```
    # 完整窗口已知匹配，返回全长度（fallback）
    return len(self.output_ids)


def _check_str_based_finish(self, new_accepted_len: int = 1):
    if (
        len(self.sampling_params.stop_strs) > 0
        or len(self.sampling_params.stop_regex_strs) > 0
    ):
        tail_str = self.tail_str(new_accepted_len)
        # 检查 stop 字符串
        if len(self.sampling_params.stop_strs) > 0:
            for stop_str in self.sampling_params.stop_strs:
                stop_str_in_tail = stop_str in tail_str
                if stop_str_in_tail or stop_str in self.decoded_text:
                    self.finished_reason = FINISH_MATCHED_STR(matched=stop_str)
                    if stop_str_in_tail:
                        # 设置精确的 finished_len，确保后续 token 被截断
                        self.finished_len = self._locate_str_stop_finished_len(
                            new_accepted_len, stop_str=stop_str
                        )
                    return True
```

python/sglang/srt/managers/detokenizer_manager.py

修改 trim_matched_stop 方法，正确处理 no_stop_trim 参数，是输出截断的配套变更。

```
def trim_matched_stop(
    self, output: Union[str, List[int]], finished_reason: Dict, no_stop_trim: bool
):
    if not finished_reason:
        return output

    matched = finished_reason.get("matched", None)
    if not matched:
        return output

    # 处理 stop 字符串
    if isinstance(matched, str) and isinstance(output, str):
        pos = output.find(matched)
        if pos == -1:
            return output
        end = pos + len(matched)
        # no_stop_trim=True 时保留 stop 字符串，False 时移除 stop 及之后内容
        return output[:end] if no_stop_trim else output[:pos]

    # 处理 stop token
    if isinstance(matched, int) and isinstance(output, list):
        if no_stop_trim:
            return output
```

```
        # 特殊处理 gpt-oss 工具调用 token
        if output[-1] == 200012 and self.is_tool_call_parser_gpt_oss:
            return output
        assert len(output) > 0
        return output[:-1]

    return output
```

评论区精华

在 review 中，gemini-code-assist[bot] 建议优化 _locate_str_stop_finished_len 的循环起始点：由于完整窗口已知匹配且旧前缀已检查过，只需从包含新接受 token 的前缀开始搜索，可显著减少 decode 调用。开发者采纳该建议并实现。最终由 hnyls2002 批准合并。

- _locate_str_stop_finished_len 性能优化 (performance): 开发者采纳建议并实现。

风险与影响

- 风险：
  1. 推测解码核心路径变更：修改了 schedule_batch.py 中的 finish 检查逻辑，可能影响非 spec 路径，但测试覆盖了 new_accepted_len=1 的 non-spec 场景。
  2. detokenizer 逻辑变更：trim_matched_stop 行为变化（尤其 no_stop_trim），需确保与上游调用兼容。
  3. 性能影响：新增 decode 调用可能带来微小开销，但通过循环优化已控制。
  4. 兼容性：旧版本中多 token 接受后输出可能包含 stop 后 token，此修复会改变行为，但这是正确的。 - 影响：影响所有使用 stop 字符串或 stop regex 的请求，特别是启用推测解码并接受多 token 的场景。对正确性至关重要，性能影响极小。团队需更新可能依赖之前错误输出行为的测试。新增单元测试可在 CPU CI 中运行，确保回归。 - 风险标记：推测解码路径变更，输出截断逻辑变更，no_stop_trim 行为变化

关联脉络

- PR #23802 fix: stop-string check misses early matches during speculative decoding: 本 PR 是 #23802 的增强，在早期匹配基础上进一步确保输出边界正确截断。