PR #28496 完整报告

sgl-project/sglang

[Spec] Fix return_hidden_states under spec V2 (issue #26163)

合并时间：2026-06-18 07:21

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/28496

执行摘要

- 一句话：修复 Spec V2 下 return_hidden_states 截断问题
- 推荐动作：PR 值得精读，尤其是对理解 speculative decoding 中张量布局和步幅切片的开发者。设计决策（如使用 num_correct_drafts_per_req_cpu + 1 作为接受长度、在 output_streamer 中二次截断）体现了对流水线中不同阶段语义的把握。建议合并后补充覆盖 EAGLE3 等算法的测试。

功能与动机

关联 Issue #26163 报告了在 speculative decoding 下 return_hidden_states=True 时，返回的 hidden states 数量少于 completion_tokens，且可能跨请求错位。问题根本在于 batch_result_processor 在 Spec V2 路径中，错误地将步幅张量 logits_output.hidden_states（形状为 [bs * speculative_num_draft_tokens, hidden_dim]）当作普通张量（形状 [bs, hidden_dim]）处理，导致每步只取 bonus token 而非整个接受序列。此修复是 V1 路径已清理后（PR #28129/#28133）的 V2 唯一跟进。

实现拆解

1. batch_result_processor 中修复 hidden states 切片（process_batch_result_decode）：在 python/sglang/srt/managers/scheduler_components/batch_result_processor.py 中，当 req.return_hidden_states 且 batch.spec_algorithm 非 none 时，不再简单取 logits_output.hidden_states[i]，而是使用 result.speculative_num_draft_tokens 作为步幅，计算每个请求的起始位置 i * stride，并根据 result.num_correct_drafts_per_req_cpu[i] + 1 确定接受长度，正确切片并 extend 到 req.hidden_states。非 Spec 路径保持不变。
2. output_streamer 中按 finished_len 截断（accept 方法）：在 python/sglang/srt/managers/scheduler_components/output_streamer.py 中，当 return_hidden_states 时，不再直接添加完整的 req.hidden_states，而是根据 req.finished_len 截断到实际完成 token 数。这是因为最后一个 verify step 的 accept_len 可能超出 max_new_tokens，导致 hidden_states 数量多于 completion_tokens。
3. 新增回归测试（test_eagle_hidden_states.py）：在 test/registered/spec/eagle/ 下新增测试文件，启动 EAGLE V2 Engine 并设置 enable_return_hidden_states=True，使用两条长度不同的 prompt 验证每请求的 hidden_states 长度等于 completion_tokens，并检查 decode row 的维度和类型正确性。测试已注册 CI（stage base-b，1-gpu-large，预计 120 秒）。

关键文件：

- python/sglang/srt/managers/scheduler_components/batch_result_processor.py（模块 调度器；类别 source；类型 core-logic；符号 process_batch_result_decode）：核心修复文件：修改了 process_batch_result_decode 中 hidden_states 的收集逻辑，正确按步幅和接受长度切片。
- python/sglang/srt/managers/scheduler_components/output_streamer.py（模块 输出流；类别 source；类型 core-logic；符号 accept）：辅助修复：在 output_streamer 的 accept 方法中，按 req.finished_len 截断 hidden_states，防止最后一个 verify step 的 accept_len overshoot max_new_tokens。
- test/registered/spec/eagle/test_eagle_hidden_states.py（模块 测试；类别 test；类型 test-coverage；符号 TestEagleReturnHiddenStates, setUpClass, tearDownClass, test_hidden_states_length_matches_completion）：新增回归测试，覆盖 EAGLE V2 下两个不同长度 prompt，验证 hidden_states 长度与 completion_tokens 一致，同时检查 decode row 类型和维度正确性。

关键符号：process_batch_result_decode, accept

关键源码片段

python/sglang/srt/managers/scheduler_components/batch_result_processor.py

核心修复文件：修改了 process_batch_result_decode 中 hidden_states 的收集逻辑，正确按步幅和接受长度切片。

```
if req.return_hidden_states and logits_output.hidden_states is not None:
    if batch.spec_algorithm.is_none():
        # Non-spec path: hidden_states is [bs, hidden_dim], directly index i
        req.hidden_states.append(
            logits_output.hidden_states[i].cpu().clone().tolist()
        )
    else:
        # Spec V2: hidden_states is [bs * speculative_num_draft_tokens, hidden_dim]
        # Each request occupies a stride of speculative_num_draft_tokens rows
        stride = result.speculative_num_draft_tokens
        # accept_len = number of correct drafts + 1 (bonus token)
        accept_len = result.num_correct_drafts_per_req_cpu[i] + 1
        start = i * stride
        # Slice the correct range for this request and extend (not append)
        req.hidden_states.extend(
            logits_output.hidden_states[start : start + accept_len]
            .cpu()
            .clone()
            .tolist()
        )
```

python/sglang/srt/managers/scheduler_components/output_streamer.py

辅助修复：在 output_streamer 的 accept 方法中，按 req.finished_len 截断 hidden_states，防止最后一个 verify step 的 accept_len overshoot max_new_tokens。

```
        if self.return_hidden_states:
            if req.return_hidden_states:
                # Mirror output_ids_through_stop: spec verify steps can overshoot finished_len.
                hs = req.hidden_states
                if req.finished_len is not None:
                    # Trim to actual finished length to avoid extra tokens
                    hs = hs[: req.finished_len]
                self.output_hidden_states.append(hs)
            else:
                self.output_hidden_states.append(None)
```

test/registered/spec/eagle/test_eagle_hidden_states.py

新增回归测试，覆盖 EAGLE V2 下两个不同长度 prompt，验证 hidden_states 长度与 completion_tokens 一致，同时检查 decode row 类型和维度正确性。

```
"""Regression test for issue #26163: return_hidden_states under EAGLE spec V2."""

import unittest
import sglang as sgl
from sglang.test.ci.ci_register import register_cuda_ci
from sglang.test.test_utils import (
    DEFAULT_DRAFT_MODEL_EAGLE,
    DEFAULT_TARGET_MODEL_EAGLE,
    CustomTestCase,
)

register_cuda_ci(est_time=120, stage="base-b", runner_config="1-gpu-large")

class TestEagleReturnHiddenStates(CustomTestCase):
    @classmethod
    def setUpClass(cls):
        # Init engine with EAGLE V2 spec and enable return_hidden_states
        cls.engine = sgl.Engine(
            model_path=DEFAULT_TARGET_MODEL_EAGLE,
            speculative_algorithm="EAGLE",
            speculative_draft_model_path=DEFAULT_DRAFT_MODEL_EAGLE,
            speculative_num_steps=3,
            speculative_eagle_topk=4,
            speculative_num_draft_tokens=8,
            enable_return_hidden_states=True,
            mem_fraction_static=0.7,
            attention_backend="triton",
        )

    @classmethod
    def tearDownClass(cls):
```

```
        if hasattr(cls, "engine") and cls.engine is not None:
            cls.engine.shutdown()

    def test_hidden_states_length_matches_completion(self):
        # Two prompts of different lengths to exercise cross-request stride aliasing
        prompts = [
            "Repeat: the quick brown fox the quick brown fox the quick brown fox",
            "Count down from ten: ten nine eight",
        ]
        max_new_tokens = 32
        outputs = self.engine.generate(
            prompts,
            sampling_params={"temperature": 0, "max_new_tokens": max_new_tokens},
            return_hidden_states=True,
        )

        for out in outputs:
            meta = out["meta_info"]
            hs = meta["hidden_states"]
            ct = meta["completion_tokens"]
            # Fixed invariant: len(hidden_states) should equal completion_tokens
            self.assertEqual(
                len(hs),
                ct,
                f"len(hidden_states)={len(hs)} but completion_tokens={ct}",
            )
            # hs[0] is the prefill block; hs[1:] are decode rows
            decode_rows = hs[1:]
            self.assertGreater(len(decode_rows), 0)
            hidden_dim = len(decode_rows[0])
            self.assertGreater(hidden_dim, 0)
            for row in decode_rows:
                self.assertIsInstance(row, list)
                self.assertEqual(len(row), hidden_dim)

if __name__ == "__main__":
    unittest.main()
```

评论区精华

Review 无评论，直接 approve。但 PR body 中作者 kpham-sgl 明确指出了问题根因并设计了修复方案。关联 Issue 中用户报告了详细的复现步骤和根因分析。整个讨论集中在正确理解 Spec V2 的 hidden_states 张量布局（步幅 vs 单行）以及如何避免 overshoot。

- 暂无高价值评论线程

风险与影响

- 风险：

1. 回归风险：核心逻辑变更涉及两个关键路径（batch_result_processor 和 output_streamer），非 Spec 路径修改较小，但 Spec V2 路径改动较大。测试覆盖了 EAGLE V2，但未覆盖 EAGLE3、STANDALONE、FROZEN_KV_MTP、DFLASH 等算法（PR body 提到的受影响路径）。
2. 性能风险：新增了切片和列表操作，但仅在 return_hidden_states=True 时生效，对正常推理路径无影响。
3. 兼容性风险：变更向后兼容，因为隐藏状态长度校正仅影响输出，不改变模型行为。 - 影响：影响范围：所有使用 Spec V2（EAGLE V2、EAGLE3、STANDALONE、FROZEN_KV_MTP、DFLASH）且开启 return_hidden_states=True 的用户。修复后，用户将获得正确的、与 completion_tokens 数量一致的 hidden states。影响程度：重要 bug 修复，解决了 silent truncation 和跨请求错位问题。非 Spec 用户不受影响。 - 风险标记：Spec V2 路径变更，缺少其他 Spec 算法测试覆盖

关联脉络

- PR #26163 [Bug] return_hidden_states truncates output under speculative decoding: 本 PR 正是为了修复该 Issue 报告的 Spec V2 路径下的 hidden_states 截断问题。
- PR #28129 Unrelated: remove V1 hidden states path: PR body 提到 Spec V1 路径已在 PR #28129 中移除，本 PR 是 V2-only 的 follow-up。
- PR #28133 Unrelated: remove V1 hidden states path: 与 #28129 类似，清理 V1 路径，确保本 PR 只处理 V2。