PR #24192 完整报告

sgl-project/sglang

[spec decoding] add tests for chain-style multi layer eagle + return_logprob

合并时间：2026-05-01 16:48

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24192

执行摘要

- 一句话：新增 chain MTP + return_logprob 测试
- 推荐动作：建议关注该测试方法的设计模式：通过两轮请求（decode vs. prefill）对比 logprobs，是一种系统性验证 spec decode 正确性的手段。此测试可作为类似功能测试的参考。

功能与动机

验证 chain MTP + spec v2 时的 logprobs 是否正确。从测试文档字符串可见，目标是确认 spec v2 decode logprobs 与 prefill scoring logprobs 匹配，确保多层 EAGLE 正确计算 logprobs。

实现拆解

1. 在 test_step3p5_flash_chain_mtp.py 中导入 numpy。
2. 新增 test_logprob_spec_v2_match 方法，使用 return_logprob 参数获取两种场景下的 logprobs。
3. 先 flush cache，对每个 prompt 发起 generate 请求（温度 0，max_new_tokens=32），返回 decode 阶段的 output_token_logprobs、output_top_logprobs 和 output_token_ids_logprobs。
4. 将 prompt 和生成的 token 拼接成完整序列，再次发起 generate 请求（max_new_tokens=0），返回 prefill-only 阶段的 input_token_logprobs 等（从 num_prompt_tokens 之后的切片）。
5. 断言两轮 logprobs 长度一致，计算 chosen token logprob 的最大绝对差异，阈值 0.255。
6. 类似地验证 top_logprobs 和 token_ids_logprob（使用宽松阈值）。使用 subTest 区分不同 prompt 以便定位失败。
7. 注册了 register_cuda_ci 以在 CI 中运行。

关键文件：

- test/registered/8-gpu-models/test_step3p5_flash_chain_mtp.py（模块 推测解码；类别 test；类型 test-coverage；符号 test_logprob_spec_v2_match）：唯一变更文件，新增了验证 chain MTP + return_logprob 正确性的测试方法。

关键符号：test_logprob_spec_v2_match

关键源码片段

test/registered/8-gpu-models/test_step3p5_flash_chain_mtp.py

唯一变更文件，新增了验证 chain MTP + return_logprob 正确性的测试方法。

```
  def test_logprob_spec_v2_match(self):
      """Verify spec v2 decode logprobs match prefill scoring logprobs.

      Generate tokens with chain MTP spec v2, then score the same sequence
      via prefill-only (no speculation). The two sets of logprobs should be
      close, validating that spec v2 + multi-layer EAGLE computes logprobs
      correctly.
      """
      requests.get(self.base_url + "/flush_cache")

      top_k = 5
      probe_token_ids = [1, 2, 10, 100, 1000]
      prompts = [
          "The capital of France is",
          "Explain quantum computing in simple terms:",
      ]

      for round_idx, prompt in enumerate(prompts):
          with self.subTest(round=round_idx, prompt=prompt):
              # 第一轮：使用 speculate decode 生成并返回 logprobs
              gen_res = requests.post(
                  self.base_url + "/generate",
                  json={
                      "text": prompt,
                      "sampling_params": {
                          "temperature": 0,
                          "max_new_tokens": 32,
                          "ignore_eos": True,
                      },
                      "return_logprob": True,
                      "top_logprobs_num": top_k,
                      "token_ids_logprob": probe_token_ids,
                      "logprob_start_len": 0,
                  },
              ).json()

              decode_logprobs = gen_res["meta_info"]["output_token_logprobs"]
              decode_top_logprobs = gen_res["meta_info"]["output_top_logprobs"]
              decode_tid_logprobs = gen_res["meta_info"]["output_token_ids_logprobs"]
              input_token_ids = [
                  t[1] for t in gen_res["meta_info"]["input_token_logprobs"]
              ]
              output_token_ids = [t[1] for t in decode_logprobs]
              num_prompt_tokens = gen_res["meta_info"]["prompt_tokens"]
```

```
# 第二轮：使用相同序列进行 prefill-only 评分（max_new_tokens=0）
score_res = requests.post(
    self.base_url + "/generate",
    json={
        "input_ids": input_token_ids + output_token_ids,
        "sampling_params": {
            "temperature": 0,
            "max_new_tokens": 0,
        },
        "return_logprob": True,
        "top_logprobs_num": top_k,
        "token_ids_logprob": probe_token_ids,
        "logprob_start_len": 0,
    },
).json()

score_logprobs = score_res["meta_info"]["input_token_logprobs"][
    num_prompt_tokens:
]
score_top_logprobs = score_res["meta_info"]["input_top_logprobs"][
    num_prompt_tokens:
]
score_tid_logprobs = score_res["meta_info"]["input_token_ids_logprobs"][
    num_prompt_tokens:
]

# 确保两轮 logprobs 数量一致
self.assertEqual(len(decode_logprobs), len(score_logprobs))

# 计算每个位置 chosen token logprob 的最大差异
decode_vals = np.array([t[0] for t in decode_logprobs])
score_vals = np.array([t[0] for t in score_logprobs])
max_diff = np.max(np.abs(decode_vals - score_vals))
print(
    f"[round {round_idx}] prompt={prompt!r} "
    f"logprob max_diff={max_diff:.6f}"
)
print(f"[round {round_idx}] decode_vals[-5:]={decode_vals[-5:]}")
print(f"[round {round_idx}] score_vals[-5:]={score_vals[-5:]}")
self.assertLess(max_diff, 0.255)

# 对于 top_logprobs 和 token_ids_logprob，由于 TP=8 和分布式噪声，
# 采用宽松阈值进行校验（具体断言代码略，但模式类似）
```

评论区精华

无实质性 design review 讨论，PR 作者自行合并。仅在 Issue 评论中触发了 CI rerun 命令。

- 暂无高价值评论线程

风险与影响

- 风险：风险较低，因为仅新增测试，不修改生产代码。需注意测试依赖 8GPU H200 环境，CI 资源消耗增加；阈值 0.255 可能在不同精度或分布式环境下需要调整。
- 影响：对用户无直接影响，但提高了 spec v2 + chain MTP 功能 logprob 正确性的测试覆盖，增强开发信心。
- 风险标记：测试覆盖，CI 资源消耗

关联脉络

- 暂无明显关联 PR