PR #26235 完整报告

sgl-project/sglang

[perf][spec decoding] Skip full-vocab softmax in EAGLE draft when topk == 1

合并时间：2026-05-25 17:06

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26235

执行摘要

- 一句话：topk==1 时跳过 full-vocab softmax
- 推荐动作：值得合并的优化：
  - 逻辑简单清晰，正确性由数学等价性保证。
  - 提供了详细的 Profile 数据和吞吐基准，说服力强。
  - 建议后续添加针对 topk==1 路径的单元测试，防止未来重构引入退化。

功能与动机

贪心投机解码路径（--speculative-eagle-topk 1）在每个草稿步和 _draft_extend_for_decode 中执行了不必要的 full-vocab torch.softmax + torch.max。Profile 显示 cunn_SoftMaxForward 每次调用约 43 µs，每个周期内触发 4 次，合计 ~175 µs/cycle 的浪费。当 topk==1 时，草稿树为单路径，topk_p 在下游不被使用（参见 spec_utils._select_top_k_tokens_later，分数沿单一分支传递），因此软最大值本身是纯粹的计算浪费。

实现拆解

1. eagle_worker_v2.py:draft_forward：在草稿步循环中，将原来的 softmax + fast_topk 替换为条件分支：当 self.topk == 1 时，直接对 logits_output.next_token_logits 执行 torch.argmax，并构造一个常数的 topk_p = ones；否则保持原逻辑。该路径运行在 DRAFT_DECODE CUDA Graph 内部。
2. eagle_worker_v2.py:_draft_extend_for_decode：在草稿扩展后重新组织 spec info 时，同样根据 self.topk == 1 条件分支，选择 argmax 或 softmax+fast_topk，消除第二个调用点的 softmax。
3. eagle_draft_extend_cuda_graph_runner.py:capture_one_batch_size：在 CUDA Graph 捕获的 run_once 闭包中，使用相同条件分支替换 softmax+fast_topk，避免 softmax 被烤进 DRAFT_EXTEND 计算图内。
4. 测试与日志：变更未引入新测试，但提供了详细的 Profile 和吞吐数据（Kimi-K2.5-NVFP4 / TP=4 / 80K ctx / EAGLE3 3-step），GPU 端 cunn_SoftMaxForward 计数在 DRAFT_DECODE 和 DRAFT_EXTEND 中均降为 0，accept_length 保持不变。

关键文件：

- python/sglang/srt/speculative/eagle_worker_v2.py（模块 投机解码；类别 source；类型 core-logic；符号 draft_forward, _draft_extend_for_decode）：核心变更文件，包含两个热度最高的调用点：draft_forward 和 _draft_extend_for_decode，分别对应草稿步循环和草稿扩展后处理。改动量为 +18/-4。
- python/sglang/srt/speculative/eagle_draft_extend_cuda_graph_runner.py（模块 投机解码；类别 source；类型 core-logic；符号 capture_one_batch_size）：第三个调用点，负责 DRAFT_EXTEND CUDA Graph 捕获阶段。改动量为 +8/-2，确保 softmax 不会被烤进 CUDA Graph 中。

关键符号：draft_forward, _draft_extend_for_decode, capture_one_batch_size

关键源码片段

python/sglang/srt/speculative/eagle_worker_v2.py

核心变更文件，包含两个热度最高的调用点：draft_forward 和 _draft_extend_for_decode，分别对应草稿步循环和草稿扩展后处理。改动量为 +18/-4。

```
# python/sglang/srt/speculative/eagle_worker_v2.py (partial)

# 在 draft_forward 方法内部，每步草稿推理后：
maybe_detect_nan(logits_output.next_token_logits, f"draft_forward step {i}")
if self.topk == 1:
    # topk=1 → 单路径退化树，topk_p 下游未使用，跳过 softmax
    topk_index = torch.argmax(
        logits_output.next_token_logits, dim=-1, keepdim=True
    )
    topk_p = torch.ones_like(topk_index, dtype=torch.float32)
else:
    probs = torch.softmax(logits_output.next_token_logits, dim=-1)
    topk_p, topk_index = fast_topk(probs, self.topk, dim=-1)

# 在 _draft_extend_for_decode 方法内：
if self.topk == 1:
    ret_topk_index = torch.argmax(
        draft_logits_output.next_token_logits, dim=-1, keepdim=True
    )
    ret_topk_p = torch.ones_like(ret_topk_index, dtype=torch.float32)
else:
    probs = torch.softmax(draft_logits_output.next_token_logits, dim=-1)
    ret_topk_p, ret_topk_index = fast_topk(probs, self.topk, dim=-1)
```

python/sglang/srt/speculative/eagle_draft_extend_cuda_graph_runner.py

第三个调用点，负责 DRAFT_EXTEND CUDA Graph 捕获阶段。改动量为 +8/-2，确保 softmax 不会被烤进 CUDA Graph 中。

```
# python/sglang/srt/speculative/eagle_draft_extend_cuda_graph_runner.py (partial)

def run_once():
```

```
# ... 前处理 ...
ret = self.model_runner.model.forward(
    forward_batch.input_ids,
    forward_batch.positions,
    forward_batch,
)
if self.topk == 1:
    # topk=1 时 argmax 等价于 softmax+argmax，且更高效
    ret.topk_index = torch.argmax(
        ret.next_token_logits, dim=-1, keepdim=True
    )
    ret.topk_p = torch.ones_like(ret.topk_index, dtype=torch.float32)
else:
    probs = torch.softmax(ret.next_token_logits, dim=-1)
    ret.topk_p, ret.topk_index = fast_topk(probs, self.topk, dim=-1)
# ... 后处理 ...
return ret
```

评论区精华

review 中无实质性讨论；gemini-code-assist[bot] 给出了自动化代码审查结论确认变更合理，kpham-sgl 给予了 APPROVAL。未发现争议或未解决问题。

- 暂无高价值评论线程

风险与影响

- 风险：低风险。变更仅限于 `self.topk == 1` 分支，不影响 topk > 1 的多路径树行为。`argmax` 与 `softmax+argmax` 在数学上等价，不会改变正确性。但应注意：
  - 如果未来逻辑修改使得 topk_p 在 topk==1 时也被用于某些计算（如概率加权），则此优化需要重新评估。
  - 缺少专门的正确性测试覆盖（从 diff 和文件列表看，test/ 目录无变更），但 PR 提供了实际模型上的 keep-accept-length 验证。
  - 影响：影响范围：仅限于 EAGLE 投机解码且 --speculative-eagle-topk 1 的配置。此配置是贪心解码的常用设置，因此对生产环境中使用 EAGLE 贪心推理的用户有直接收益。影响程度：中等。在 Kimi-K2.5 模型上，1000/Mean TPOT 提升约 2.1%（+8.8 tok/s），延迟降低约 0.05 ms。影响面较小，但收益稳定且无副作用。
- 风险标记：缺少测试覆盖，核心路径变更

关联脉络

- PR #26241 [perf][spec decoding] Skip common_template in TRTLLMMLAMultiStepDraftBackend init: 同属 speculative-decoding 性能优化系列，针对 EAGLE/Trtllm MLA 路径。
- PR #26244 [Spec] fix EAGLE v2 verify metadata init order on non-cuda-graph path: 同文件 eagle_worker_v2.py 的近期 bugfix，修复了 EAGLE 相关元数据初始化顺序。