PR #32334 完整报告

sgl-project/sglang

[Speculative Decoding] Fix GPT-OSS EAGLE3 hidden states

合并时间：2026-08-01 02:58

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/32334

执行摘要

- 一句话：修复 GPT-OSS EAGLE3 隐藏状态捕获与归一化错误
- 推荐动作：值得精读。这是一个典型的 " 为新模型修正既有约定 " 的 bugfix：代码量小（36 行），但修复路径完整呈现了从崩溃定位、语义分析、兼容性权衡到多模型基准验证的全过程。特别值得关注两个设计决策：一是把捕获语义显式定义为 boundary 0 / boundary i+1 并重构循环，使最终边界与中间边界统一；二是在无法从配置判断约定时，选择 max(layer_ids) == num_layers 这一最小侵入的向后兼容规则，而非更宽泛的启发式。建议后续为 EAGLE3 层 ID 语义补充自动化测试，避免类似偏移问题再次回归。

功能与动机

issue #32226 报告：服务 openai/gpt-oss-120b 搭配 nvidia/gpt-oss-120b-Eagle3-v3 时引擎在 warmup 即崩溃，报 RuntimeError: The size of tensor a (8640) must match the size of tensor b (5760)；手动把 eagle_aux_hidden_state_layer_ids 里的 36 改成 34 可以绕开崩溃，但 MT-Bench 接受率仅 1.1，而 TRT-LLM 同配置可达 2.6。PR body 给出根因：SGLang 将边界 [24, 30, 36] 偏移成 [25, 31, 37] 且没有捕获所有 36 个 GPT-OSS block 之后的最终边界，所以只返回两个 2880 宽 state；同时 draft 配置的 norm_before_fc: true 与其 input_norm.weight 被忽略，导致即使不崩溃接受率也偏低。

实现拆解

实现按 4 步推进，全部集中在模型实现层，不涉及 kernel 或调度：

1. 重构 GPT-OSS 捕获循环（python/sglang/srt/models/gpt_oss.py 的 GptOssTransformer.forward）。旧实现是在处理 block i 之前检查 i in self.layers_to_capture 并捕获 hidden_states + residual，这导致边界 num_hidden_layers（最终归一化前的最后一个输出）在循环内永远无法被观察到。新实现改为：循环前捕获 boundary 0（即 embedding 输出），每个 block 处理完成后检查 i + 1 in self.layers_to_capture 再捕获该 block 的输出。这样最终边界与中间边界走同一条路径，删除了此前对 end_layer 特殊分支的需求。注意 set_dflash_layers_to_capture 仍保留独立的 +1 转换，因为 DFlash 的 ID 是零基 block 索引。
2. 修正 EAGLE3 layer ID 转换（GptOssForCausalLM.set_eagle3_layers_to_capture）。旧代码无条件执行 [val + 1 for val in layer_ids]。新代码先取 num_layers = self.config.num_hidden_layers，若 layer_ids 非空且 max(layer_ids) == num_layers，说明配置里已经包含最终隐藏状态边界、ID 本身就是边界索引，直接 list(layer_ids) 使用；否

则保留历史 +1 转换。这样 v3 的 [24, 30, 36] 不再变成 [25, 31, 37]，而 long-context 的 [1, 17, 32] 仍然转成 [2, 18, 33]，保持既有行为。

3. 支持 norm_before_fc（python/sglang/srt/models/llama_eagle3.py 的 EAGLE3 模型类）。在初始化时从 eagle_config 读取 norm_before_fc（回退到顶层 config 属性），启用时为 hidden_size_in * num_aux_hidden_states 创建统一 RMSNorm input_norm；forward 中在进入 fc 之前先对整个拼接向量做归一化。它与已有 fc_norm 的差异是：fc_norm 对每个特征 chunk 独立归一化后再拼接，而 norm_before_fc 对拼接后的完整向量做一次归一化，两者语义不同、可叠加。
4. 测试与验证配套。本 PR 最初新增了两个单测文件（test_gpt_oss_eagle3_capture.py、test_llama_eagle3_norm_before_fc.py），在 review 中应 kpham-sgl 要求删除，回归验证依靠注册的模型测试（test_gpt_oss_120b.py）与 CI 重跑；验收以 2x B200 / 2x H200 上的完整 80 题 MT-Bench 为基准，PR 描述补充了四个 EAGLE3 头的 BEFORE/AFTER 对比表。

关键文件：

- python/sglang/srt/models/gpt_oss.py（模块 模型实现；类别 source；类型 core-logic；符号 GptOssTransformer.forward, GptOssForCausalLM.set_eagle3_layers_to_capture）：核心修复文件：重构隐藏状态边界捕获循环（boundary 0 与 boundary i+1 统一路径），并在 set_eagle3_layers_to_capture 中加入 max(layer_ids) == num_layers 时跳过 +1 的兼容判断，解决 v3 头崩溃根因。
- python/sglang/srt/models/llama_eagle3.py（模块 草稿模型；类别 source；类型 core-logic；符号 LlamaEagle3ForCausalLM.init, LlamaEagle3ForCausalLM.forward）：新增 norm_before_fc / input_norm 支持：为拼接后的完整目标特征向量在 FC 投影前应用 RMSNorm，解决 draft 配置中 norm_before_fc: true 被忽略导致接受率偏低的问题。

关键符号：GptOssTransformer.forward, GptOssForCausalLM.set_eagle3_layers_to_capture, LlamaEagle3ForCausalLM.init, LlamaEagle3ForCausalLM.forward

关键源码片段

python/sglang/srt/models/gpt_oss.py

核心修复文件：重构隐藏状态边界捕获循环（boundary 0 与 boundary i+1 统一路径），并在 set_eagle3_layers_to_capture 中加入 max(layer_ids) == num_layers 时跳过 +1 的兼容判断，解决 v3 头崩溃根因。

```
# GptOssTransformer.forward() 中的隐藏状态边界捕获（修复后）
# 语义约定：boundary 0 为 embedding 输出；boundary i + 1 为第 i 个 transformer block 的输出。
# 旧实现是在 block i 处理前采样，导致最后一个边界（num_hidden_layers）永远无法被观察到。
aux_hidden_states = []

# boundary 0：即第一个 block 的输入（embedding 输出）
if self.start_layer in self.layers_to_capture:
    aux_hidden_states.append(
        hidden_states + residual if residual is not None else hidden_states
    )
```

```
for i in range(self.start_layer, self.end_layer):
    with get_global_expert_distribution_recorder().with_current_layer(i):
        layer = self.layers[i]
        hidden_states, residual = layer(positions, hidden_states, forward_batch, residual)
        # 每个 block 处理完后检查边界，最终边界与中间边界走同一路径
        if i + 1 in self.layers_to_capture:
            aux_hidden_states.append(
                hidden_states + residual if residual is not None else hidden_states
            )

def set_eagle3_layers_to_capture(self, layer_ids: Optional[List[int]] = None):
    if not self.pp_group.is_last_rank:
        return

    num_layers = self.config.num_hidden_layers
    if layer_ids is None:
        self.capture_aux_hidden_states = True
        self.model.layers_to_capture = [2, num_layers // 2, num_layers - 3]
    else:
        self.capture_aux_hidden_states = True
        # 若 max(layer_ids) == num_layers，说明配置中已包含最终隐藏状态边界，
        # 即 ID 本身就是边界索引，直接使用，避免 +1 后越界；
        # 否则保留历史的 +1 转换（把零基 block ID 转为 block 后边界）。
        if layer_ids and max(layer_ids) == num_layers:
            self.model.layers_to_capture = list(layer_ids)
        else:
            self.model.layers_to_capture = [val + 1 for val in layer_ids]
```

python/sglang/srt/models/llama_eagle3.py

新增 norm_before_fc / input_norm 支持：为拼接后的完整目标特征向量在 FC 投影前应用 RMSNorm，解决 draft 配置中 norm_before_fc: true 被忽略导致接受率偏低的问题。

```
# EAGLE3 draft 前端初始化：拼接目标模型特征后先过 FC 投影
eagle_config = getattr(config, 'eagle_config', None) or {}

# norm_before_fc：对拼接后的完整特征向量统一做一次 RMSNorm 再进 FC；
# 注意与 fc_norm 的区别：fc_norm 是对每个特征 chunk 独立归一化后再拼接。
self.norm_before_fc = bool(
    eagle_config.get('norm_before_fc', getattr(config, 'norm_before_fc', False))
)
if self.norm_before_fc:
    self.input_norm = RMSNorm(
        self.hidden_size_in * self.num_aux_hidden_states,
        eps=config.rms_norm_eps,
    )
else:
    self.input_norm = None

# forward() 中应用归一化：先整体、再逐 chunk，最后进 FC
hidden_states = forward_batch.spec_info.hidden_states
```

```
if hidden_states.shape[-1] != embeds.shape[-1]:
    # 先应用 input_norm（整个拼接向量），再按需应用逐 chunk 的 fc_norm
    if self.input_norm is not None:
        hidden_states = self.input_norm(hidden_states)
    if self.fc_norm is not None:
        chunks = hidden_states.chunk(self.num_aux_hidden_states, dim=-1)
        hidden_states = torch.cat(
            [norm(chunk) for norm, chunk in zip(self.fc_norm, chunks)],
            dim=-1,
        )
    hidden_states = self.fc(hidden_states)
```

评论区精华

1. +1 偏移的由来（design，已解决）：nvpohanh 追问 +1 的动机，Codex 追查到 #8824——GPT-OSS 复制了 Llama 的捕获循环，旧循环在 block i 前采样，因此零基 block ID 需要 +1 转成 post-block 边界。修复后把语义显式改为边界索引。
2. end_layer 特殊处理（design，已解决）：nvpohanh 问为何要特殊处理 end_layer，Codex 解释旧逻辑在 block 前捕获导致最后一个边界无法观察，重构为循环前捕获 boundary 0、block 后捕获 boundary i+1 后不再需要特殊分支。
3. 四变体回归验证（testing，已解决）：nvpohanh 要求确认 long-context / short-context / throughput 三个变体仍正常并汇总 AL 表，Codex 在 2x H200 上完成完整 MT-Bench 验证并按要求更新了 PR 描述的 BEFORE/AFTER 表。
4. 单测删除（testing，已解决）：kpham-sgl 表示 "We don't need these unit tests"，Codex 在 commit 88d69e56ae 中删除了两个新增单测。
5. 兼容规则收敛（correctness，已解决）：kpham-sgl 建议参考 #25454，用 max(layer_ids) == num_layers 判断是否跳过 +1（"don't do the +1 ... if max(layers_id) == num_layers"）；Codex 先尝试 layer_ids[0] == 1 的启发式，被 nvpohanh 质疑（" Why does 'layer_ids[0] == 1' implies that the configs follows the first convention? The config could be following the first convention while having layer_ids[0] = 2"），最终收敛为仅按 max(layer_ids) == num_layers 判断。

- +1 层 ID 偏移的由来与正确语义 (design): 重构捕获语义：boundary 0 为 embedding 输出，block i 后捕获 boundary i+1，EAGLE3 ID 按边界语义解释。
- end_layer 特殊处理的必要性 (design): 重构后每个 block 后即检查边界，最终边界与中间边界统一路径，删除特殊分支。
- 四个 EAGLE3 头变体的回归验证 (testing): 四个头全部通过，PR 描述补充 BEFORE/AFTER 对比表，BEFORE v3 行记录预期崩溃。
- norm_before_fc 语义与注释 (documentation): 在初始化处补充内联注释。
- 单元测试是否保留 (testing): Codex 在 commit 88d69e56ae 中删除两个测试文件，回归交由注册模型测试覆盖。
- max(layer_ids)==num_layers 兼容规则的收敛 (correctness): 采用 max(layer_ids) == num_layers 才跳过 +1，v3 保持 [24, 30, 36]，long-context 保持 [2, 18, 33]。

风险与影响

- 风险：
  1. 捕获循环重构的回归风险：GptOssTransformer.forward 的捕获点从 block 前移到 block 后，对既有 +1 配置而言语义等价（边界 i+1 本就是 block i 的输出），风险较低；但 DFlash 路径仍沿用旧函数 set_dflash_layers_to_capture，虽然其 +1 后不可能等于 num_layers 从而不受 max 判断影响，仍需注意两套转换逻辑并存容易再次漂移。
  2. 启发式兼容规则的风险：max(layer_ids) == num_layers 只是经验规则，若未来出现输出层 ID 恰好包含 num_layers 的新配置，会错误跳过 +1 导致捕获错误；nvpohanh 自己也指出配置中并无显式约定标记。
  3. 自动化测试覆盖不足：新单测被删除后，该逻辑回归只能依赖 test_gpt_oss_120b.py 等注册模型测试，而这些测试未必覆盖 EAGLE3 layer ID 语义的边界情况。
  4. norm_before_fc 与 fc_norm 叠加顺序：实现中 input_norm 先于 fc_norm 应用，若某配置同时启用两者，需要与参考实现核对归一化顺序是否一致。 - 影响：对用户：gpt-oss-120b + Eagle3-v3 从启动即崩溃变为可用，接受率恢复至 2.72，输出吞吐从不可用提升到 843.79 tok/s；long-context / short-context / throughput 三个变体的行为保持一致（long-context 恢复至 2.325 的历史基线），吞吐均有小幅提升。对系统：改动局限在 gpt_oss.py 与 llama_eagle3.py 两个模型实现文件，不涉及 kernel、调度器或 KV cache 等核心路径，对其他模型无影响。对团队：明确了 EAGLE3 层 ID 存在两种约定（输出层 ID vs 捕获边界 ID）及兼容处理模式，为后续类似模型接入提供了可复用的判断范式。 - 风险标记：核心路径变更，启发式兼容规则，缺少自动化测试覆盖

关联脉络

- PR #25454 EAGLE3 layer ID 约定相关 PR（review 中引用，标题未提供）：kpham-sgl 在 review 中明确引用该 PR：它定义了 EAGLE3 层 ID 的两种约定（输出层 ID vs 捕获边界 ID），本 PR 的 max(layer_ids)==num_layers 兼容规则正是对其思路的简化沿用。
- PR #31430 Remove unused draft-extend CUDA graph top-k: 同属 GPT-OSS / EAGLE3 speculative decoding 路径的清理与优化，体现该功能线的持续演进。
- PR #32690 [Fix] missing max_context_len on HybridAttnBackend: 同为 EAGLE verify 崩溃类 bugfix（attention 后端缺配置导致 verify 崩溃），与本 PR 的修复目标一致，保证 speculative decoding 路径的稳定性。