

PR #34459 完整报告

sgl-project/sglang

Fix DeepSeek-V4 routing: sqrtsoftplus underflow and unfloored renorm

合并时间: 2026-09-10 15:10

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/34459>

执行摘要

- 一句话: 修复 DeepSeek-V4 路由器 sqrtsoftplus 下溢和权重归一化除零, 防止专家权重 NaN。
- 推荐动作: 该 PR 是一个高质量的、针对性的 bugfix, 解决了可能导致生产事故的严重数值问题。作者在 PR body 中提供了极其详尽的技术分析, 展示了对问题的深刻理解。由于其修复了核心路由路径的数值稳定性, 且变更涉及 Triton/CUDA 内核, 强烈建议所有关注模型推理稳定性和 MoE 路由正确性的工程师精读。特别关注其实现的 log1p 数学技巧和归一化 epsilon 的添加方式。虽然没有讨论线程, 但其技术方案清晰合理。

功能与动机

DeepSeek-V4 的 sqrtsoftplus 门控有四种路由实现, 每种都缺失了参考门控 (`inference/model.py Gate.forward`, 使用 `F.softplus`) 提供的一部分数值保护。关联的 issue #30989 和 flashinfer-ai/flashinfer#3803 报告了类似的 NaN 问题。具体而言:

1) Triton 路由器 (默认, 第 3-42 层) 在 fp32 下, 当 $x \leq -16.64$ 时, $\log(1.0 + \exp(x))$ 会下溢为 0。若所有 top-K 专家都低于此阈值, 行权重归一化后为全零, 令牌将丢失其整个路由专家贡献。2) `hash_topk` (第 0-2 层) 在 CUDA 内核 (默认) 及其 torch 回退中, 除以 `routed_sum` 时没有添加 epsilon 保护。哈希路由忽略分数, 因此全下溢行会导致 $0/0 = \text{NaN}$ 。

实现拆解

变更在四个文件中修复了两类核心数值缺陷。

1. Triton 路由器 softplus 数值稳定性修复: 修改 `python/sglang/kernels/ops/moe/moe_fused_gate.py` 中的 Triton 内核 `_router_triton_kernel`。将原先不稳定的 `tl.log(1.0 + tl.exp(scores))` 计算, 替换为使用 log1p 数学技巧的精确计算: `tl.where(exact, z, z * tl.log(u) / tl.where(exact, 1.0, u - 1.0))`, 其中 $z = \exp(-\text{abs}(x))$, $u = 1 + z$, 并结合 `tl.maximum(scores, 0.0)` 进行符号折叠。此修改确保了在 $[-104, 200]$ 全范围内, 相对误差从 1.0 降低到 $3.5e-07$ 。
2. 归一化 epsilon 添加: 在以下三处添加了极小的归一化 epsilon, 以防止除零: a. `hash_topk` 的 torch 回退: 修改 `python/sglang/srt/layers/moe/hash_topk.py` 的 `_forward_torch` 方法, 在两处 `topk_weights` 归一化除法中, 为分母添加 `_RENORMALIZE_SUM_EPSILON` (从 `sglang.srt.layers.moe.topk` 导入)。这覆盖了 `num_fused_shared_experts == 1` 和 `!= 1` 两种分支。b. `hash_topk` 的 CUDA 内核: 修

改 `python/sglang/kernels/jit/csrc/deepseek_v4/hash_topk.cuh`, 在 `moe_hash_topk_fused` 内核中, 将权重归一化从 `routed_weight / routed_sum` 改为 `routed_weight / (routed_sum + kRenormalizeSumEpsilon)`, 其中 `kRenormalizeSumEpsilon = 1e-20f`。c. CUDA `moe_fused_gate` 的 `softplus` 修复: 修改 `python/sglang/kernels/jit/csrc/moe/moe_fused_gate.cuh` 中的 `compute_score` 函数, 将 `log1pf(expf(x))` 替换为与 Triton 内核相同的符号折叠公式: `fmaxf(x, 0.0f) + log1pf(expf(-fabsf(x)))`, 以解决溢出问题。

本 PR 仅包含核心逻辑修改, 未涉及测试文件变更, 但 PR body 声明已通过 `test_moe_fused_gate.py` 的 39 个测试。

关键文件:

- `python/sglang/kernels/ops/moe/moe_fused_gate.py` (模块 MoE 路由; 类别 source; 类型 core-logic) : 包含默认 Triton 路由器的核心逻辑, 是 `sqrtsoftplus` 计算的主要修复点, 修复了下溢和溢出问题。
- `python/sglang/srt/layers/moe/hash_topk.py` (模块 MoE 路由; 类别 source; 类型 core-logic) : 包含 `hash_topk` 的 torch 回退实现, 修复了归一化除零导致 NaN 的问题。
- `python/sglang/kernels/jit/csrc/deepseek_v4/hash_topk.cuh` (模块 MoE 路由; 类别 other; 类型 core-logic) : 包含 `hash_topk` 的 CUDA 内核实现, 修复了归一化除零问题, 与 torch 回退逻辑保持一致。
- `python/sglang/kernels/jit/csrc/moe/moe_fused_gate.cuh` (模块 MoE 路由; 类别 other; 类型 core-logic) : 包含 CUDA 版本的 `moe_fused_gate` 路由器, 修复了 `softplus` 计算溢出问题。

关键符号: `HashTopK._forward_torch`, `compute_score`, `moe_hash_topk_fused`

关键源码片段

`python/sglang/kernels/ops/moe/moe_fused_gate.py`

包含默认 Triton 路由器的核心逻辑, 是 `sqrtsoftplus` 计算的主要修复点, 修复了下溢和溢出问题。

修复前: 在 Triton 路由器内核中, `log(1.0 + exp(x))` 在 `x` 较小时下溢, 在 `x` 较大时溢出。

`sp = tl.where(scores > 20.0, scores, tl.log(1.0 + tl.exp(scores)))`

修复后: 使用 `log1p` 的精确数学技巧计算

`softplus`, 并进行符号折叠以避免大负数下溢和大正数溢出。

`log1p(z) = z*log(u)/(u-1)` where `z=exp(-|x|)`, `u=1+z`。

这样计算精确且数值稳定, 在 `[-104, 200]` 范围内最大相对误差仅为 `3.5e-07`。

`z = tl.exp(-tl.abs(scores))`

`u = 1.0 + z`

`exact = u == 1.0`

`log1p_z = tl.where(exact, z, z * tl.log(u) / tl.where(exact, 1.0, u - 1.0))`

`sp = tl.maximum(scores, 0.0) + log1p_z`

`activated = tl.sqrt(sp)`

`python/sglang/srt/layers/moe/hash_topk.py`

包含 `hash_topk` 的 torch 回退实现，修复了归一化除零导致 NaN 的问题。

```
# 修复前：直接除以路由权重之和，当和为 0 时会导致 0/0 = NaN。
# topk_weights[:, :-1] /= topk_weights[:, :-1].sum(dim=-1, keepdim=True)

# 修复后：在分母中加入一个极小的 epsilon (_RENORMALIZE_SUM_EPSILON = 1e-20)
# 防止除零。
# 这个 epsilon 非常小，对正常情况 (sum >> epsilon) 的结果没有影响，仅在 sum
# 接近零时提供保护。
if self.score_func != "softmax":
    topk_weights[:, :-1] /= (
        topk_weights[:, :-1].sum(dim=-1, keepdim=True)
        + _RENORMALIZE_SUM_EPSILON
    )
```

python/sglang/kernels/jit/csrc/deepseek_v4/hash_topk.cuh

包含 `hash_topk` 的 CUDA 内核实现，修复了归一化除零问题，与 torch 回退逻辑保持一致。

```
// 定义与 Python 侧对应的极小 epsilon 常量，用于归一化保护。
constexpr float kRenormalizeSumEpsilon = 1e-20f;

// ... 在内核函数 moe_hash_topk_fused 中 ...
// 修复前：直接除以 routed_sum，可能为 0。
// topk_weights[output_offset] = is_shared ? 1.0f / routed_scaling_factor : routed_weight /
// routed_sum;

// 修复后：在分母中添加 epsilon，防止除零。对正常情况无影响。
topk_weights[output_offset] =
    is_shared ? 1.0f / routed_scaling_factor : routed_weight / (routed_sum +
        kRenormalizeSumEpsilon);
```

评论区精华

PR 的 review 过程极为简洁，仅有一条批准记录，没有公开的讨论线程。主要技术洞察和决策都体现在作者的 PR body 和提交信息中，其中详细阐述了问题根因、数学原理和修复方案。这表明变更的正确性主要通过代码审查和作者的专业知识来保证，而非通过激烈的线上讨论。

- 暂无高价值评论线程

风险与影响

- 风险：
 1. 数值精度回归风险：修改了核心的 softplus 计算和归一化逻辑。虽然数学上与参考实现 F.softplus 对齐，且声明通过了现有测试，但涉及广泛的模型层（第 0-42 层），任何实现细微偏差都可能影响模型输出质量或训练稳定性。需要持续监控生产环境中的模型行为。
 2. 硬件兼容性风险：CUDA 和 Triton 内核被修改，影响了 GPU 后端。尽管代码本身是平台无关的，但不同 GPU 架构（如 Blackwell B300）的浮点行为可能存在细微差异，可能在特定边界条件下触发未预料到的问题。

3. 测试覆盖不足风险：PR 未添加新的针对性单元测试来验证这些特定的数值边界条件（如输入为 -200、+200）。现有的测试（如 `test_moe_fused_gate.py`）可能覆盖不全，存在回归风险。

• 影响：

1. 用户影响：直接影响所有使用 DeepSeek-V4 模型（具有 `sqrtsoftplus` 门控）的用户。修复了可能导致长上下文或特定输入下输出崩溃（NaN、全零专家权重）的严重 bug，提升了模型服务的稳定性和可靠性。
2. 系统影响：影响 SGLang 的 MoE 路由子系统，特别是默认的 Triton 路由器和 `hash_topk` 路径。是核心推理路径的关键数值修复。
3. 团队影响：为团队提供了一个处理类似数值稳定性问题的参考案例。通过统一归一化 `epsilon` 的做法（复用 `_RENORMALIZE_SUM_EPSILON`），有助于代码库内的一致性和维护。
- 风险标记：核心路径变更，数值精度敏感，缺少新增测试覆盖

关联脉络

- PR #30989 [Bug] GLM-5.2-NVFP4 + flashinfer_trtllm: long-context outputs collapse to '!!!!' (NaN logits) — fp32 correction_bias regression from #29783: 本 PR 的直接灵感来源。该 issue 报告了 DeepSeekV3 路由中类似的 NaN 问题（归一化除零），本 PR 将其修复模式推广并应用于 DeepSeek-V4 的 `sqrtsoftplus` 门控实现。
- PR #3803 Fix 0/0 NaN in GLM52 routing renorm on sigmoid underflow: 关联 issue，描述了同类问题（sigmoid 下溢导致归一化 NaN）的修复。本 PR 的修复与此 issue 解决的是同一类数值稳定性问题，但针对不同的门控函数和模型。