PR #26349 完整报告

sgl-project/sglang

Support specific pass of bias_grouped_topk for xpu

合并时间：2026-06-03 13:13

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26349

执行摘要

- 一句话：XPU 端为 MoE 门控添加 bias group topk 快速路径
- 推荐动作：该 PR 功能明确，讨论均已解决，测试已补充，建议合并。值得关注的设计决策是 num_experts 上限 256 的设定和 scaling 处理与 CUDA 分支的对称性。

功能与动机

PR body 指出：'Use topk_sigmoid directly when num_groups is 1, just like cuda did.' 目的是对齐 CUDA 已有优化，在 XPU 上绕过复杂的通用实现，加速 MoE 门控计算。

实现拆解

实现分为以下步骤：

1. 添加 XPU 快速分支：在 biased_grouped_topk_gpu 函数（python/sglang/srt/layers/moe/topk.py）中新增 *elif (_is_xpu and num_expert_group == 1 and topk_group == 1 and ...)* 分支，直接调用 topk_sigmoid 计算 topk 权重和索引，并将结果乘以缩放因子 scaling 后返回。
2. 缩放因子处理：与 CUDA 分支相同，检查 apply_routed_scaling_factor_on_output：若为 False，则将 scaling 强制置为 1.0，避免不期望的缩放。
3. 空 token 边界：在 num_tokens == 0 时提前返回预分配的空张量，与 CUDA 分支行为一致。
4. 条件约束：当前 XPU 分支限制为 num_experts <= 256（CUDA 分支为 512），这是因为 XPU 上的 topk_sigmoid 实现可能对专家数量更敏感，后续可酌情放宽。
5. 单元测试：新增 test/registered/xpu/test_topk.py，包含 TestBiasedGroupedTopK 类，复用 Nemotron-3-Nano-30B-A3B 的配置（E=128, G=1, topk=6）测试多种 token 数量（1024/2048/4096/8192），对比 biased_grouped_topk_gpu 与原生 biased_grouped_topk_impl 结果是否一致。

关键文件：

- python/sglang/srt/layers/moe/topk.py（模块 MoE 路由；类别 source；类型 *core-logic*；符号 biased_grouped_topk_gpu）：核心修改：在 biased_grouped_topk_gpu 函数中新增 XPU 快速路径分支，直接调用 topk_sigmoid 替代通用实现。

- test/registered/xpu/test_topk.py（模块 MoE 路由；类别 test；类型 test-coverage；符号 TestBiasedGroupedTopK, _run_single_test, test_fast_biased_grouped_topk）：新增 XPU 偏置分组 topk 的单元测试，涵盖 Nemotron 模型配置，验证 fused 实现与原生实现一致。

关键符号：biased_grouped_topk_gpu

关键源码片段

python/sglang/srt/layers/moe/topk.py

核心修改：在 biased_grouped_topk_gpu 函数中新增 XPU 快速路径分支，直接调用 topk_sigmoid 替代通用实现。

```
elif (
    _is_xpu
    and num_expert_group == 1
    and topk_group == 1
    and num_fused_shared_experts == 0
    and num_experts <= 256 # XPU 上限低于 CUDA（512），由硬件特性决定
    and topk <= 8
):
    # 当 apply_routed_scaling_factor_on_output 为 False 时，禁用 scaling
    if not apply_routed_scaling_factor_on_output:
        scaling = 1.0

    num_tokens = gating_output.shape[0]

    # 预分配输出张量，避免动态分配开销
    topk_values = torch.empty(
        (num_tokens, topk), dtype=torch.float32, device=gating_output.device
    )
    topk_indices = torch.empty(
        (num_tokens, topk), dtype=torch.int32, device=gating_output.device
    )

    # 空序列场景直接返回
    if num_tokens == 0:
        return topk_values, topk_indices

    # 调用 topk_sigmoid 进行核心计算
    topk_sigmoid(
        topk_values,
        topk_indices,
        gating_output,
        renormalize,
        correction_bias,
    )
    return topk_values * scaling, topk_indices
```

test/registered/xpu/test_topk.py

新增 XPU 偏置分组 topk 的单元测试，涵盖 Nemotron 模型配置，验证 fused 实现与原生实现一致。

```
import unittest
import torch

from sglang.srt.layers.moe.topk import (
    biased_grouped_topk_gpu,
)
from sglang.srt.layers.moe.topk import (
    biased_grouped_topk_impl as native_biased_grouped_topk,
)
from sglang.test.ci.ci_register import register_xpu_ci
from sglang.test.test_utils import CustomTestCase

# 注册到 XPU CI，估计时间 5s，归入 stage-b-test-1-gpu-xpu 套件
register_xpu_ci(est_time=5, suite="stage-b-test-1-gpu-xpu")


# Nemotron-3 使用 biased_grouped_topk
class TestBiasedGroupedTopK(CustomTestCase):
    def _run_single_test(self, M, E, G, topk, topk_group, renormalize,
                         gating_dtype, bias_dtype, routed_scaling_factor):
        torch.manual_seed(1024)
        device = torch.device("xpu")

        # 注意：hidden_states 在 bfloat16 下若值太小可能全相等，因此用 randn(M,100) 保证多样性
        hidden_states = torch.randn(M, 100, dtype=torch.bfloat16, device=device)
        gating_output = torch.randn(M, E, dtype=gating_dtype, device=device)
        correction_bias = torch.randn(E, dtype=bias_dtype, device=device)

        # 原生实现作为参考
        ref_topk_weights, ref_topk_ids = native_biased_grouped_topk(
            hidden_states, gating_output, correction_bias,
            topk, renormalize, G, topk_group,
            routed_scaling_factor=routed_scaling_factor,
        )

        # fused 版本（即新增的 XPU 快速路径）
        topk_weights, topk_ids = biased_grouped_topk_gpu(
            hidden_states, gating_output, correction_bias,
            topk, renormalize, G, topk_group, 0,
            routed_scaling_factor, None,
        )

        # 将结果展开为 (M, E) 全表后进行 close 比较
        res = torch.zeros(M, E, dtype=torch.float, device=device)
```

```
        ref = torch.zeros(M, E, dtype=torch.float, device=device)
        res.scatter_(1, topk_ids.long(), topk_weights)
        ref.scatter_(1, ref_topk_ids.long(), ref_topk_weights)
        torch.testing.assert_close(res, ref)

    def test_fast_biased_grouped_topk(self):
        # 配置来自 Nemotron-3-Nano-30B-A3B 模型
        E_num = 128
        num_expert_group = 1
        topk_value = 6
        topk_group = 1
        gating_dtype = torch.bfloat16
        bias_dtype = torch.float32
        renormalize = True
        routed_scaling_factor = 2.5

        bs = [1, 2, 4, 8]
        seq_len = 1024
        num_tokens = [b * seq_len for b in bs] # [1024, 2048, 4096, 8192]

        for M in num_tokens:
            self._run_single_test(
                M, E_num, num_expert_group, topk_value, topk_group,
                renormalize, gating_dtype, bias_dtype, routed_scaling_factor,
            )


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

评论区精华

Review 中主要讨论两点：

1. Scalding 处理与测试覆盖：mingfeima 询问是否需要处理 apply_routed_scaling_factor_on_output，并建议补充测试。gaopengff 回复已添加 scaling 检查，测试会在模型就绪后补充（实际本 PR 已包含测试）。
2. CI 时间估计调整：mingfeima 指出 est_time=300 过大，会影响 CI 任务划分，建议改为实际运行时间。gaopengff 随后改为 5，并表示测试实际很快。

- apply_routed_scaling_factor_on_output 处理与测试更新 (design): 作者已添加 scaling 检查（if not apply_routed_scaling_factor_on_output: scaling = 1.0），并且已在当前 PR 中补充测试。
- est_time 参数设置 (testing): gaopengff 将 est_time 改为 5，认为测试实际很快。

风险与影响

- 风险：

1. 硬件特定条件可能遗漏场景：XPU 分支限制 num_experts<=256，若未来模型使用更多专家将 fallback 到通用路径，虽不影响正确性但失去优化。
2. 测试覆盖有限：仅测试 Nemotron 配置下的正确性，未覆盖边界值（如最大专家数、varying topk）或 non-trivial scaling 场景。
3. 与 CUDA 分支行为差异：CUDA 分支使用 jit_grouped_topk，XPU 使用 topk_sigmoid，两者底层实现不同，若数值误差可能产生微小的输出差异，但测试已经对比原生实现，风险低。
4. 潜在回归风险：修改发生在 MoE 核心路由路径，若出现未预料的 XPU 条件会走通用路径，不影响其他硬件。 - 影响：对 XPU 用户，使用符合快速路径条件的 MoE 模型（如 Nemotron）将获得性能提升；对其它硬件无影响。团队需要维护两个硬件路径的条件逻辑，带来少量维护成本。对于 CI，新增的 XPU 测试以 est_time=5 注册在 stage-b-test-1-gpu-xpu 套件中，影响轻微。 - 风险标记：硬件特定优化条件，条件分支覆盖面，测试覆盖有限

关联脉络

- PR #25655 Feat/add w4a16 moe support to nemotron: Nemotron 模型使用了 biased_grouped_topk，该 PR 在 XPU 上优化了同一路由，是功能延续。