PR #29761 完整报告

sgl-project/sglang

[Bugfix] compressed-tensors WNA16 MoE: don't assume a "Linear" config group

合并时间：2026-07-02 02:08

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/29761

执行摘要

- 一句话：修复 compressed-tensors WNA16 MoE 加载无 Linear 组的检查点时 KeyError
- 推荐动作：该 PR 修复了一个重要的加载阻塞 bug，建议阅读者关注其设计模式：将每层 weight_quant 的解析逻辑收敛在 get_moe_scheme 中，而不是分散在各方案构造函数中。这符合 DRY 原则，也便于未来扩展新的 MoE 量化方案。新增的回归测试同样值得参考。

功能与动机

CompressedTensorsWNA16MoE.__init__读取 target_scheme_map["Linear"]，对于 config_groups 使用 regex 或每层目标（例如混合精度 INT4/INT8 MoE）且没有命名为 "Linear" 的组的检查点，会抛出 KeyError: 'Linear'。测试在混合精度 int4/int8 MoE 检查点上验证修复前加载失败，修复后正常加载并生成。

实现拆解

1. 修改 WNA16 MoE 方案构造函数（compressed_tensors_wNa16_moe.py）：添加 weight_quant: QuantizationArgs 参数，替换原本从 self.quant_config.target_scheme_map["Linear"] 读取 weights 的逻辑，直接从 weight_quant 获取每层权重量化参数。
2. 修改 MxInt4 MoE 方案构造函数（compressed_tensors_w4a4_mxint4_moe.py）：同样添加 weight_quant 参数并替换 target_scheme_map["Linear"] 查找。
3. 更新调用起点（compressed_tensors.py）：在 get_moe_scheme 方法中，将已经通过层名匹配解析的 weight_quant 显式传递给各个 MoE 方案构造函数（包括 WNA16 Marlin、Triton 和 MxInt4），而不是仅传入 self。
4. 添加回归测试（test/.../test_compressed_tensors_wna16_moe_no_linear.py）：构造没有 "Linear" 组的 MoE 量化配置（使用 regex 或每层 FQN 目标），断言 get_moe_scheme 正确返回 WNA16 方案且不抛出异常。测试运行在 CPU，无权重创建和 kernel 执行。

关键文件：

- python/sglang/srt/layers/quantization/compressed_tensors/schemes/compressed_tensors_wNa16_moe.py（模块 量化层；类别 source；类型 core-logic；符号 init）：核心修复文件：修改 __init__方法，添加 weight_quant 参数并移除 target_scheme_map['Linear'] 查找。

- test/registered/unit/layers/quantization/test_compressed_tensors_wna16_moe_no_linear.py（模块 测试；类别 test；类型 test-coverage；符号 _make_wna16_moe_config, TestWNA16MoENoLinearGroup, _assert_wna16_moe, test_regex_expert_targets_int4）：新增回归测试文件，测试没有 Linear 组的 MoE 配置能否正确解析方案，覆盖 INT4/INT8、regex/FQN 目标变体。
- python/sglang/srt/layers/quantization/compressed_tensors/compressed_tensors.py（模块 量化层；类别 source；类型 core-logic）：调用层修改：在 get_moe_scheme 中将已解析的 weight_quant 传递给各个 MoE 方案构造函数。
- python/sglang/srt/layers/quantization/compressed_tensors/schemes/compressed_tensors_w4a4_mxint4_moe.py（模块 量化层；类别 source；类型 core-logic；符号 init）：与 WNA16 方案同步修改，确保 MxInt4 MoE 方案也通过参数接收 weight_quant。

关键符号：CompressedTensorsWNA16MoE.init, CompressedTensorsMxInt4MoE.init, CompressedTensorsConfig.get_moe_scheme

关键源码片段

python/sglang/srt/layers/quantization/compressed_tensors/schemes/compressed_tensors_wNa16_moe.py

核心修复文件：修改 __init__方法，添加 weight_quant 参数并移除 target_scheme_map['Linear'] 查找。

```
from __future__ import annotations
# ...（其他导入省略）

class CompressedTensorsWNA16MoE(CompressedTensorsMoEScheme):

    def __init__(
        self,
        quant_config: CompressedTensorsConfig,
        weight_quant: QuantizationArgs, # 新增参数：由 get_moe_scheme
        预先解析的每层权重量化参数
        num_gpu_experts: int = -1,
    ):
        self.quant_config = quant_config
        # 替换原有 self.quant_config.target_scheme_map["Linear"].get("weights")
        # 混合精度 MoE 检查点可能没有 "Linear" 组，使用已解析的 weight_quant 直接赋值
        config = weight_quant
        self.num_bits = config.num_bits
        self.packed_factor = 32 // config.num_bits
        self.strategy = config.strategy
        self.group_size = config.group_size
        self.actorder = config.actorder
        self.sym = config.symmetric

        if not (
            self.quant_config.quant_format == CompressionFormat.pack_quantized.value
```

```
            and self.num_bits in WNA16_SUPPORTED_BITS
        ):
            raise ValueError(
                "For Fused MoE layers, only ",
                f"{CompressionFormat.pack_quantized.value} ",
                "is supported for the following bits: ",
                f"{WNA16_SUPPORTED_BITS}",
            )
        self.num_gpu_experts = num_gpu_experts

    # ... ( 其他方法不变 )
```

test/registered/unit/layers/quantization/test_compressed_tensors_wna16_moe_no_linear.py

新增回归测试文件，测试没有 Linear 组的 MoE 配置能否正确解析方案，覆盖 INT4/INT8、regex/FQN 目标变体。

```
"""CPU 回归测试：WNA16 compressed-tensors MoE 在没有 'Linear' 组时能正常工作"""

from sglang.srt.layers.quantization.compressed_tensors.compressed_tensors import (
    CompressedTensorsConfig,
)
from sglang.srt.layers.quantization.compressed_tensors.schemes import (
    CompressedTensorsWNA16MoE,
    CompressedTensorsWNA16TritonMoE,
)
from sglang.test.test_utils import CustomTestCase

# 允许的 WNA16 MoE 方案（Marlin 或 Triton 后端均可）
_WNA16_MOE_SCHEMES = (CompressedTensorsWNA16MoE,
CompressedTensorsWNA16TritonMoE)
EXPERTS_LAYER = "model.layers.0.mlp.experts"


def _make_wna16_moe_config(targets, num_bits):
    """构造不含 'Linear' 组的 WNA16 MoE 量化配置"""
    return {
        "quant_method": "compressed-tensors",
        "format": "pack-quantized",
        "config_groups": {
            "group_0": {
                "targets": targets,
                "weights": {
                    "num_bits": num_bits,
                    "type": "int",
                    "symmetric": True,
                    "strategy": "group",
                    "group_size": 128,
                },
```

```
                "input_activations": None,
            }
        },
        "ignore": ["lm_head", "re:.*self_attn.*", "re:.*mlp.gate$"],
    }


class TestWNA16MoENoLinearGroup(CustomTestCase):
    """回归测试：确保没有 'Linear' 组时也能正确解析 MoE 方案"""

    def _assert_wna16_moe(self, config_dict, expected_bits):
        quant_config = CompressedTensorsConfig.from_config(config_dict)
        # 验证没有 "Linear" 组（预条件，否则就失去了测试意义）
        self.assertNotIn("Linear", quant_config.target_scheme_map)

        layer = torch.nn.Module()
        # 调用 get_moe_scheme —— 修复前会抛出 KeyError: 'Linear'
        scheme = quant_config.get_moe_scheme(layer, layer_name=EXPERTS_LAYER)

        self.assertIsInstance(scheme, _WNA16_MOE_SCHEMES)
        self.assertEqual(scheme.num_bits, expected_bits)
        self.assertEqual(scheme.group_size, 128)

    def test_regex_expert_targets_int4(self):
        config = _make_wna16_moe_config(["re:.*mlp.experts.*"], num_bits=4)
        self._assert_wna16_moe(config, expected_bits=4)

    def test_regex_expert_targets_int8(self):
        config = _make_wna16_moe_config(["re:.*mlp.experts.*"], num_bits=8)
        self._assert_wna16_moe(config, expected_bits=8)

    def test_per_layer_fqn_expert_targets_int4(self):
        config = _make_wna16_moe_config(
            [f"{EXPERTS_LAYER}.0.gate_proj",
             f"{EXPERTS_LAYER}.0.up_proj",
             f"{EXPERTS_LAYER}.0.down_proj"],
            num_bits=4
        )
        self._assert_wna16_moe(config, expected_bits=4)

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

评论区精华

BBuf 在代码审查中建议为没有 "Linear" 组的 MoE 配置添加回归测试，以防止 target_scheme_map["Linear"] 回退再次引入。joerowell 采纳了建议并在后续提交中添加了测试文件。

- 建议添加回归测试 (testing): joerowell 同意并添加了测试文件 test_compressed_tensors_wna16_moe_no_linear.py。

风险与影响

- 风险：风险较低。主要风险在于方案构造函数签名增加了 weight_quant 参数，但所有调用点已在 get_moe_scheme 中同步更新。如果未来有新的 MoE 方案没有相应更新，可能会导致 TypeError。但该风险通过显式参数传递和静态类型检查可降低。另外，混合精度 MoE 检查点的加载路径以前未经过充分测试，新增的回归测试覆盖了主要变体（INT4/INT8、regex/FQN 目标），减少了回归风险。
- 影响：直接影响使用 compressed-tensors 量化且包含 MoE 层的用户，特别是使用混合精度（如部分 INT4 部分 INT8）或通过 regex/per-layer 目标指定配置组而没有 "Linear" 组的检查点。之前这些用户会遇到 KeyError 无法加载模型，现在可以正常加载。其他用户（如使用均匀量化和有 "Linear" 组的检查点）不受影响。
- 风险标记：内部 API 变更，需要同步其他 MoE 方案

关联脉络

- 暂无明显关联 PR