PR #46860 完整报告

vllm-project/vllm

[Bugfix][Quantization] Fix W8A8 int-quantized scheme selection regression

合并时间：2026-06-29 23:55

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/46860

执行摘要

- 一句话：修复 W8A8 int8 静态量化方案选择回归
- 推荐动作：建议立即合入。修复简洁明确，测试覆盖完整（含正向和反向验证），端到端验证通过。设计启示：量化方案路由条件应通过参数化测试严格覆盖，类似回归可通过条件分支变更规范化审查避免。

功能与动机

PR #46389 在重构时误将 is_static_int8_in and is_static_int8_out 改为 or，导致静态输入量化但无输出量化的 W8A8 int8 模型被错误归类为 W8A8O8，引起推理结果错误。本 PR 恢复原始条件并补充测试防止回归。

实现拆解

1. 核心修复：在 vllm/model_executor/layers/quantization/compressed_tensors/compressed_tensors.py 的 _is_wNa8o8_int 方法中，将返回条件从 is_static_int8_in or is_static_int8_out 恢复为 is_static_int8_in and is_static_int8_out。只有输入和输出均满足静态 INT8 要求时，才会选用 WNA8O8Int 假量化方案，否则进入其他 int8 GEMM 路径。
2. 测试增强：在 tests/quantization/test_compressed_tensors.py 中添加 196 行参数化测试，定义 _STATIC_SYM_INT8_ACT、_STATIC_ASYM_INT8_ACT、_DYNAMIC_INT8_ACT 等量化参数常量，并通过 13 个 pytest.param 用例覆盖 W8A8（channel/tensor weight、symmetric/asymmetric/dynamic input）、W8A8O8、W4A8O8 以及 pack-quantized（2-8 bit）等组合，每个用例调用 _get_scheme_from_parts 并断言返回的方案类。
3. 验证：测试中特意包含 2 个依赖回归前条件（即 or 行为）但应走 W8A8Int8 的用例，在老代码上会失败，确认回归检测有效。同时通过 vllm serve nm-testing/w8a8_static_asym-e2e 验证端到端正确性。

关键文件：

- vllm/model_executor/layers/quantization/compressed_tensors/compressed_tensors.py（模块 量化器；类别 source；类型 core-logic；符号 _is_wNa8o8_int）：核心修复文件，修改 _is_wNa8o8_int 方法中的条件从 or 恢复为 and，仅此一行变更解决了 W8A8 int8 静态模型的方案选择回归。

- tests/quantization/test_compressed_tensors.py（模块 测试；类别 test；类型 test-coverage；符号 test_scheme_selection）：新增 196 行全面参数化测试，覆盖回归点（W8A8 int8 static 无 output_quant）及多种量化配置，含反向验证（老代码失败），确保方案选择逻辑正确。

关键符号：_is_wNa8o8_int, test_scheme_selection

关键源码片段

vllm/model_executor/layers/quantization/compressed_tensors/compressed_tensors.py

核心修复文件，修改 _is_wNa8o8_int 方法中的条件从 or 恢复为 and，仅此一行变更解决了 W8A8 int8 静态模型的方案选择回归。

```
  def _is_wNa8o8_int(
      self,
      weight_quant: QuantizationArgs,
      input_quant: QuantizationArgs | None,
      output_quant: QuantizationArgs | None,
      format: str | None,
  ) -> bool:
      """Weight N-bit INT (pack-quantized for sub-byte, int-quantized for 8-bit)
      with static per-tensor INT8 input/output activation quant, applied as a float
      fake-quant around a weight-only matmul."""
      is_int_pack_format = format in (
          CompressionFormat.pack_quantized.value,
          CompressionFormat.int_quantized.value,
      )
      is_channel_group = weight_quant.strategy in (
          QuantizationStrategy.CHANNEL.value,
          QuantizationStrategy.GROUP.value,
      )
      is_static_int = (
          weight_quant.type == QuantizationType.INT and not weight_quant.dynamic
      )
      is_intN_weight = is_static_int and is_channel_group and is_int_pack_format

      is_static_int8_in = (
          input_quant is not None
          and input_quant.type == QuantizationType.INT
          and input_quant.strategy == QuantizationStrategy.TENSOR.value
          and input_quant.num_bits == 8
          and not input_quant.dynamic
      )
      is_static_int8_out = (
          output_quant is not None
          and output_quant.type == QuantizationType.INT
          and output_quant.strategy == QuantizationStrategy.TENSOR.value
          and output_quant.num_bits == 8
```

```
        and not output_quant.dynamic
    )
    # 修正：只有 input 和 output 同时为静态 INT8 才走 W8A8O8 路径
    # 此前 PR#46389 误改为 or，导致仅 input 静态时也被路由至 W8A8O8
    return is_intN_weight and (is_static_int8_in and is_static_int8_out)
```

tests/quantization/test_compressed_tensors.py

新增 196 行全面参数化测试，覆盖回归点（W8A8 int8 static 无 output_quant）及多种量化配置，含反向验证（老代码失败），确保方案选择逻辑正确。

```
# 预定义激活量化常量
_STATIC_SYM_INT8_ACT = QuantizationArgs(
    num_bits=8, type=QuantizationType.INT,
    strategy=QuantizationStrategy.TENSOR.value,
    symmetric=True, dynamic=False,
)
_STATIC_ASYM_INT8_ACT = QuantizationArgs(
    num_bits=8, type=QuantizationType.INT,
    strategy=QuantizationStrategy.TENSOR.value,
    symmetric=False, dynamic=False,
)
_DYNAMIC_INT8_ACT = QuantizationArgs(
    num_bits=8, type=QuantizationType.INT,
    strategy=QuantizationStrategy.TOKEN.value,
    symmetric=True, dynamic=True,
)

@pytest.mark.parametrize(
    "weight_bits,weight_strategy,input_act,output_act,format,expected_scheme",
    [
        # 回归点：W8A8 int-quantized 静态输入，无输出量化 → W8A8Int8
        pytest.param(8, QuantizationStrategy.CHANNEL.value, _STATIC_SYM_INT8_ACT, None,
                     "int-quantized", CompressedTensorsW8A8Int8,
                     id="w8a8_channel_static_sym"),
        pytest.param(8, QuantizationStrategy.CHANNEL.value, _STATIC_ASYM_INT8_ACT, None,
                     "int-quantized", CompressedTensorsW8A8Int8,
                     id="w8a8_channel_static_asym"),
        pytest.param(8, QuantizationStrategy.TENSOR.value, _STATIC_SYM_INT8_ACT, None,
                     "int-quantized", CompressedTensorsW8A8Int8,
                     id="w8a8_tensor_static"),
        pytest.param(8, QuantizationStrategy.CHANNEL.value, _DYNAMIC_INT8_ACT, None,
                     "int-quantized", CompressedTensorsW8A8Int8,
                     id="w8a8_channel_dynamic"),
        # W8A8O8: input 和 output 都是静态 INT8 → CompressedTensorsWNA8O8Int
        pytest.param(8, QuantizationStrategy.CHANNEL.value, _STATIC_SYM_INT8_ACT, _
        STATIC_SYM_INT8_ACT,
                     "int-quantized", CompressedTensorsWNA8O8Int,
                     id="w8a8o8_channel"),
        pytest.param(4, QuantizationStrategy.GROUP.value, _STATIC_SYM_INT8_ACT, _STATIC_
```

```
        SYM_INT8_ACT,
                "int-quantized", CompressedTensorsWNA8O8Int,
                id="w4a8o8_group"),
        # pack-quantized → CompressedTensorsWNA16
        pytest.param(8, QuantizationStrategy.CHANNEL.value, None, None,
                "pack-quantized", CompressedTensorsWNA16,
                id="w8_pack"),
        pytest.param(4, QuantizationStrategy.GROUP.value, None, None,
                "pack-quantized", CompressedTensorsWNA16,
                id="w4_pack"),
        pytest.param(2, QuantizationStrategy.GROUP.value, None, None,
                "pack-quantized", CompressedTensorsWNA16,
                id="w2_pack"),
        pytest.param(3, QuantizationStrategy.GROUP.value, None, None,
                "pack-quantized", CompressedTensorsWNA16,
                id="w3_pack"),
        pytest.param(5, QuantizationStrategy.GROUP.value, None, None,
                "pack-quantized", CompressedTensorsWNA16,
                id="w5_pack"),
        pytest.param(6, QuantizationStrategy.GROUP.value, None, None,
                "pack-quantized", CompressedTensorsWNA16,
                id="w6_pack"),
        pytest.param(7, QuantizationStrategy.GROUP.value, None, None,
                "pack-quantized", CompressedTensorsWNA16,
                id="w7_pack"),
        pytest.param(8, QuantizationStrategy.GROUP.value, None, None,
                "pack-quantized", CompressedTensorsWNA16,
                id="w8_group_pack"),
    ],
)
def test_scheme_selection(weight_bits, weight_strategy, input_act, output_act, format, expected_
scheme):
    """验证 _get_scheme_from_parts 选择正确的量化方案。"""
    config = _make_ct_config(target="Linear")
    weight_quant = QuantizationArgs(
        num_bits=weight_bits, type=QuantizationType.INT,
        strategy=weight_strategy, symmetric=True, dynamic=False,
    )
    scheme = config._get_scheme_from_parts(
        weight_quant=weight_quant, input_quant=input_act,
        output_quant=output_act, format=format,
    )
    assert isinstance(scheme, expected_scheme), (
        f"Expected {expected_scheme.__name__}, got {type(scheme).__name__}"
    )
```

评论区精华

无人工讨论议题。维护者 mgoin 直接批准并回复 "Thanks!"，claude[bot] 自动化审查因 fork 未启用。

- 暂无高价值评论线程

风险与影响

- 风险：低风险。修复仅涉及一行条件逻辑变更，且经过新增 13 个参数化测试和端到端模型验证。潜在风险：还原为 and 后，仅有输出量化而无输入量化的组合（极少见）将不会再被路由到 W8A8O8 路径，但此行为与原始设计意图一致，回归前即为 and。测试未覆盖此类情形，但根据代码逻辑，若不满足 and 会由后续方案处理，不会直接出错。
- 影响：修复了 PR #46389 引入的回归，影响使用 CompressedTensors W8A8 int-quantized 方案（带静态输入量化）的用户，此前可能因错误路由导致模型加载失败或推理结果异常。对 W8A8O8 用户无影响，条件更严格但逻辑正确。影响范围小且针对性明确。
- 风险标记：核心路径变更，回归修复，已补充测试覆盖

关联脉络

- PR #46389 Refactor W8A8 quant conditions（引入回归）: 本 PR 修复 #46389 引入的回归，将错误改变的 or 恢复为 and。