# PR #42543 完整报告

vllm-project/vllm

[Compilation] Skip x.size(dim) in _decompose_size_nodes

合并时间：2026-07-16 08:59

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/42543

## 执行摘要

- 一句话：跳过带 dim 的 size() 节点，修复编译时 LoRA 崩溃
- 推荐动作：值得精读。本 PR 展示了一个典型的 FX 图编译边界问题修复过程，从错误的复杂方案最终收敛到最小改动方案。审查中的讨论体现了对 torch.compile 内部与 torch.Size 概念的理解。建议关注 compilation 模块的类似边界问题。

## 功能与动机

Unsloth + vLLM 的 LoRA 微调工作负载在使用 torch.compile 时触发 RuntimeError: Tried to erase Node size_1 but it still had 2 users in the graph。原因是 _decompose_size_nodes 函数未能正确处理带 dim 参数的 x.size(0) 节点，这些节点出现在 slice 对象内部，导致分解后仍被 slice 引用，擦除时崩溃。根本原因是 _decompose_size_nodes 只应分解无 dim 参数的 x.size()（返回 torch.Size 元组），而 x.size(dim) 已返回标量，不应分解。

## 实现拆解

1. 在 vllm/compilation/backends.py 中修改 _decompose_size_nodes 函数，在遍历 size_nodes 时加入 if len(node.args) > 1 or "dim" in node.kwargs: continue，跳过带有 dim 参数的 size 调用。
2. 在 tests/compile/test_graph_partition.py 中新增 test_decompose_size_leaves_scalar_size_with_dim 回归测试，构造包含 x.size(0) 并嵌入 slice 的 FX 图，验证函数不会崩溃且 size 节点保持不变。

关键文件：

- vllm/compilation/backends.py（模块 编译后端；类别 source；类型 core-logic；符号 _decompose_size_nodes）：核心逻辑文件，增加跳过检查，是修复的主要变更
- tests/compile/test_graph_partition.py（模块 图分区测试；类别 test；类型 test-coverage；符号 test_decompose_size_leaves_scalar_size_with_dim）：新增回归测试，覆盖带 dim 的 size() 节点场景

关键符号：_decompose_size_nodes

## 关键源码片段

## vllm/compilation/backends.py

核心逻辑文件，增加跳过检查，是修复的主要变更

```
# vllm/compilation/backends.py

def _decompose_size_nodes(graph: fx.GraphModule) -> None:
    """Decompose x.size() into per-dim sym_size.int calls..."""
    size_nodes = list(graph.graph.find_nodes(op='call_method', target='size'))

    for node in size_nodes:
        # 关键修复：如果 size() 带 dim 参数（如 x.size(0)），直接跳过
        if len(node.args) > 1 or 'dim' in node.kwargs:
            continue

        tensor_node = node.args[0]
        ev = tensor_node.meta.get('example_value')
        assert ev is not None, (
            f'Tensor node "{tensor_node.name}" has no example_value metadata. '
            f'Cannot decompose size node "{node.name}".'
        )

        # Build per-dim replacements: sym_size.int node or literal int.
        dims: list[fx.Node | int] = []
        with graph.graph.inserting_after(tensor_node):
            for i in range(ev.dim()):
                dim_val = ev.shape[i]
                if isinstance(dim_val, torch.SymInt):
                    dn = graph.graph.call_function(
                        torch.ops.aten.sym_size.int, args=(tensor_node, i))
                    dn.meta['example_value'] = dim_val
                    dims.append(dn)
                elif isinstance(dim_val, int):
                    dims.append(dim_val)
                else:
                    raise AssertionError(
                        f'dim_val is either torch.SymInt or int, '
                        f'got {type(dim_val)} for dim {i} of "{node.name}"'
                    )

        # 替换用户 args 的原有逻辑未改动，此处省略
        for user in list(node.users):
            pass
```

## tests/compile/test_graph_partition.py

新增回归测试，覆盖带 dim 的 size() 节点场景

```
# tests/compile/test_graph_partition.py

def test_decompose_size_leaves_scalar_size_with_dim():
```

```
    """
    Regression test: _decompose_size_nodes must leave x.size(dim) alone.
    x.size() returns a torch.Size tuple that needs decomposition,
    but x.size(dim) returns a scalar SymInt/int that should be preserved.
    """
    from torch._dynamo.source import LocalSource
    from torch._subclasses.fake_tensor import FakeTensorMode
    from torch.fx.experimental.symbolic_shapes import ShapeEnv

    graph = fx.Graph()
    x = graph.placeholder('x')
    mapping = graph.placeholder('token_lora_mapping')
    size_node = graph.call_method('size', args=(x, 0))
    sliced_node = graph.call_function(
        operator.getitem,
        args=(mapping, slice(None, size_node, None)),
    )
    graph.output((sliced_node,))

    shape_env = ShapeEnv()
    src = LocalSource('tokens')
    sym_tokens = shape_env.create_symintnode(shape_env.create_symbol(4, src), hint=4)
    fake_mode = FakeTensorMode(shape_env=shape_env)
    with fake_mode:
        fake_x = torch.empty_strided((sym_tokens, 8), (8, 1))
    x.meta['example_value'] = fake_x

    gm = fx.GraphModule(torch.nn.Module(), graph)

    # Should not raise "Tried to erase Node size ... still had N users"
    _decompose_size_nodes(gm)

    # The scalar x.size(0) node is left in place
    remaining = list(gm.graph.find_nodes(op='call_method', target='size'))
    assert len(remaining) == 1, f'Expected 1 size node, found {len(remaining)}'
    assert remaining[0].args == (x, 0), f'Size node args changed: {remaining[0].args}'
```

## 评论区精华

zou3519 在代码审查中指出原始问题的根源是 _decompose_size_nodes 不应处理 x.size(0) 节点，因为 x.size(dim) 返回标量。他建议直接忽略带 dim 参数的 size 调用。作者接受了建议并简化了修复，同时补充了回归测试。最终 zou3519 批准了合并。

- 修复方案应跳过 x.size(dim) 而非处理 slice 嵌套 (design): 作者采纳建议，将修复简化为在循环中检查并跳过带 dim 的 size 节点。
- 添加回归测试并确认修复 (testing): 测试已添加，并与 Unsloth 工作负载验证成功。

## 风险与影响

- 风险：风险较低。改动仅 5 行，且有针对性回归测试覆盖。但需确认所有可能的 dim 参数形式都被覆盖：node.args 长度大于 1 捕获 x.size(0)，dim in node.kwargs 捕获 x.size(dim=0) 等 KWARG 形式。未覆盖的潜在情况可能包括未来 Dynamo 生成的其他变体，但当前框架内已足够。
- 影响：本 PR 修复了 LoRA 微调在 torch.compile 模式下的编译崩溃，直接影响使用 Unsloth + vLLM 进行微调的用户。对于其他不涉及 x.size(dim) 的模型，无影响。系统运行稳定性提升。
- 风险标记：核心路径变更，测试覆盖完善，低风险

## 关联脉络

- 暂无明显关联 PR