PR #28546 完整报告

sgl-project/sglang

[diffusion] Fix FP8 fused TP scale loading

合并时间：2026-06-18 14:38

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/28546

执行摘要

- 一句话：修复融合 FP8 per-tensor scale 在 TP 下加载错误
- 推荐动作：值得精读，特别是权重加载器处理特殊参数（PerTensorScaleParameter）的设计模式。为类似量化参数的加载提供了可参考的标量广播实现。

功能与动机

ModelOpt FP8 checkpoints can store fused linear per-tensor scales as scalar tensors with shape [1]. For fused MergedColumnParallelLinear / QKVParallelLinear parameters, the loader previously routed the scalar through the shard-0 path only. The remaining logical scale slots stayed uninitialized, and later quantization setup used .max() over the fused scale buffer, which could pick garbage values and corrupt FP8 TP outputs.

实现拆解

1. 核心逻辑修复：在 MergedColumnParallelLinear.weight_loader_v2 和 QKVParallelLinear.weight_loader_v2 中，当 loaded_shard_id 为 None 且参数是 PerTensorScaleParameter 时，新增两个条件判断：
   - 如果 loaded_weight 元素数为 1 且参数元素数 > 1，则用标量填充整个参数（广播到所有 fused 槽位）。
   - 否则如果形状匹配，直接复制整个张量（处理完整向量场景）。 原逻辑只在 tp_size > 1 且形状匹配时才复制，标量场景未正确处理。
2. 新增单元测试：新建 test/unit/test_parallel_linear_weight_loading.py，包含四个测试：test_merged_column_parallel_scalar_scale_load_fills_fused_slots、test_qkv_parallel_scalar_scale_load_fills_fused_slots、test_merged_column_parallel_full_scale_vector_loads_all_fused_slots、test_qkv_parallel_full_scale_vector_loads_all_fused_slots。覆盖 MergedColumnParallelLinear 和 QKVParallelLinear 的标量广播和完整向量复制。
3. 调整端到端测试：在 gpu_cases.py 中移除 ONE_GPU_MODELOPT_FP8_CASES 中的 flux2_modelopt_fp8_t2i（原 1GPU 一致性测试），并添加到 TWO_GPU_CASES 中作为 flux2_modelopt_fp8_tp2_t2i（TP2 一致性测试），以验证多卡 FP8 TP 的正确性。
4. 更新测试数据版本：在 test_utils.py 中更新 SGL_TEST_FILES_CI_DATA_REVISION 以匹配新的 benchmark ground truth。

关键文件：

- python/sglang/multimodal_gen/runtime/layers/linear.py（模块 线性层；类别 source；类型 core-logic；符号 weight_loader_v2）：核心修复逻辑，在 weight_loader_v2 中添加标量广播和形状匹配复制处理。
- python/sglang/multimodal_gen/test/unit/test_parallel_linear_weight_loading.py（模块 单元测试；类别 test；类型 test-coverage；符号 _per_tensor_scale, test_merged_column_parallel_scalar_scale_load_fills_fused_slots, test_qkv_parallel_scalar_scale_load_fills_fused_slots, test_merged_column_parallel_full_scale_vector_loads_all_fused_slots）：新增单元测试文件，验证四种 key 加载路径。
- python/sglang/multimodal_gen/test/server/gpu_cases.py（模块 GPU 测试；类别 test；类型 test-coverage；符号 ONE_GPU_MODELOPT_FP8_CASES, TWO_GPU_CASES）：将 FLUX.2 FP8 e2e 测试从 1GPU 改为 2GPU TP2 一致性测试，验证多卡修复。
- python/sglang/multimodal_gen/test/test_utils.py（模块 测试工具；类别 test；类型 test-coverage；符号 SGL_TEST_FILES_CI_DATA_REVISION）：更新 ci data revision 以匹配新的 benchmark ground truth。

关键符号：weight_loader_v2, _per_tensor_scale, test_merged_column_parallel_scalar_scale_load_fills_fused_slots, test_qkv_parallel_scalar_scale_load_fills_fused_slots, test_merged_column_parallel_full_scale_vector_loads_all_fused_slots, test_qkv_parallel_full_scale_vector_loads_all_fused_slots

关键源码片段

python/sglang/multimodal_gen/runtime/layers/linear.py

核心修复逻辑，在 weight_loader_v2 中添加标量广播和形状匹配复制处理。

```
def weight_loader_v2(
    self,
    param: BasevLLMParameter,
    loaded_weight: torch.Tensor,
    loaded_shard_id: int | None = None,
) -> None:
    if isinstance(param, BlockQuantScaleParameter):
        self._weight_loader_v2_block_quant_scale(
            param, loaded_weight, loaded_shard_id
        )
        return

    if loaded_shard_id is None:
        if isinstance(param, PerTensorScaleParameter):
            # 修复：标量 scale（numel=1）需要广播到整个 fused buffer
            if loaded_weight.numel() == 1 and param.data.numel() > 1:
                param.data.fill_(loaded_weight.reshape(-1)[0])
                return
```

```
            # 如果形状已匹配，直接复制（处理完整向量场景）
            if loaded_weight.shape == param.data.shape:
                param.data.copy_(loaded_weight)
                return
            param.load_merged_column_weight(loaded_weight=loaded_weight, shard_id=0)
            return
        elif type(param) in (RowvLLMParameter, BasevLLMParameter):
            param.load_merged_column_weight(loaded_weight=loaded_weight)
            return
        # TODO: @dsikka - move to parameter.py
        self._load_fused_module_from_checkpoint(param, loaded_weight)
        return

    assert loaded_shard_id < len(self.output_sizes)
    tp_size = self.tp_size
    shard_offset = sum(self.output_sizes[:loaded_shard_id]) // tp_size
    shard_size = self.output_sizes[loaded_shard_id] // tp_size
    param.load_merged_column_weight(
        loaded_weight=loaded_weight,
        shard_id=loaded_shard_id,
        shard_offset=shard_offset,
        shard_size=shard_size,
    )
```

python/sglang/multimodal_gen/test/unit/test_parallel_linear_weight_loading.py

新增单元测试文件，验证四种 key 加载路径。

```
import pytest
import torch
from sglang.multimodal_gen.runtime.layers.linear import (
    MergedColumnParallelLinear,
    QKVParallelLinear,
)
from sglang.multimodal_gen.runtime.models.parameter import PerTensorScaleParameter

def _per_tensor_scale(values: list[float]) -> PerTensorScaleParameter:
    return PerTensorScaleParameter(
        data=torch.tensor(values, dtype=torch.float32),
        weight_loader=lambda *_args, **_kwargs: None,
    )

@pytest.mark.parametrize("loaded_weight", [torch.tensor(0.25), torch.tensor([0.25])])
def test_merged_column_parallel_scalar_scale_load_fills_fused_slots(loaded_weight):
    layer = MergedColumnParallelLinear.__new__(MergedColumnParallelLinear)
    layer.tp_size = 2
    param = _per_tensor_scale([-1.0, -2.0])
    layer.weight_loader_v2(param, loaded_weight)
    # 标量应广播到所有 fused 槽位，结果应为 [0.25, 0.25]
```

```
    assert torch.equal(param.data, torch.tensor([0.25, 0.25]))

@pytest.mark.parametrize("loaded_weight", [torch.tensor(0.5), torch.tensor([0.5])])
def test_qkv_parallel_scalar_scale_load_fills_fused_slots(loaded_weight):
    layer = QKVParallelLinear.__new__(QKVParallelLinear)
    layer.tp_size = 2
    param = _per_tensor_scale([-1.0, -2.0, -3.0])
    layer.weight_loader_v2(param, loaded_weight)
    # 标量应广播到所有 fused 槽位，结果应为 [0.5, 0.5, 0.5]
    assert torch.equal(param.data, torch.tensor([0.5, 0.5, 0.5]))

def test_merged_column_parallel_full_scale_vector_loads_all_fused_slots():
    layer = MergedColumnParallelLinear.__new__(MergedColumnParallelLinear)
    layer.tp_size = 1
    param = _per_tensor_scale([-1.0, -2.0])
    # 加载形状匹配的向量 [0.25, 0.75]
    layer.weight_loader_v2(param, torch.tensor([0.25, 0.75]))
    assert torch.equal(param.data, torch.tensor([0.25, 0.75]))

def test_qkv_parallel_full_scale_vector_loads_all_fused_slots():
    layer = QKVParallelLinear.__new__(QKVParallelLinear)
    layer.tp_size = 1
    param = _per_tensor_scale([-1.0, -2.0, -3.0])
    layer.weight_loader_v2(param, torch.tensor([0.25, 0.5, 0.75]))
    assert torch.equal(param.data, torch.tensor([0.25, 0.5, 0.75]))
```

评论区精华

PR 由作者自行合并，无 review 讨论。仅有的两条评论来自 gemini-code-assist[bot] 提示每日配额达到上限，以及作者触发 /tag-and-rerun-ci。变更方案直接明了，社区无争议。

- 暂无高价值评论线程

风险与影响

- 风险：修复涉及核心权重加载路径（weight_loader_v2），影响所有使用 PerTensorScaleParameter 的融合线性层。逻辑限定于标量形状（numel() == 1）和形状匹配两种情况，回归风险较低。新增单元测试覆盖了主要场景，但未覆盖例如 tp_size=2 且完整向量形状匹配的情况（旧逻辑已处理）。端到端测试仅覆盖 FLUX.2 FP8，其他模型（如 Wan2.2、HunyuanVideo）未在 TP 下测试，可能存在未被发现的边界条件。
- 影响：修复直接影响使用 ModelOpt FP8 checkpoints 的扩散模型用户在张量并行（TP）下的推理正确性，特别是 FLUX.2 模型。用户不再需要绕过此问题（如使用单卡或禁用 FP8）。对未使用 TP 或非 FP8 模型的用户无影响。测试覆盖的加入降低了回归风险。
- 风险标记：核心权重加载路径，影响多 GPU FP8 模型，端到端测试仅覆盖 FLUX.2

关联脉络

- 暂无明显关联 PR