PR #32013 完整报告

sgl-project/sglang

:bug: [llm][npu][quant] Fix ModelSlim MXFP4 packed weight loading

合并时间：2026-07-29 11:34

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/32013

执行摘要

- 一句话：修复 NPU 上 ModelSlim MXFP4 打包权重加载失败
- 推荐动作：对于维护 NPU 量化管线的开发者，建议仔细审阅此数据契约变更，确保检查点与代码版本一致。Review 中关于 NZ 格式的权衡讨论值得注意。

功能与动机

PR #32013 修复了 Ascend NPU 上离线 ModelSlim W4A4_MXFP4 检查点加载失败的问题。根本原因是 ModelSlim 在导出时应用了 pack_fp4_to_uint8，而 SGLang 仍然期望每个 FP4 值单独存放在 float8_e4m3fn 中的旧布局，导致形状不匹配。

实现拆解

1. 在 modelslim_mxfp4.py 中更新数据契约：将 weight 参数形状从 [out, in] dtype float8_e4m3fn 改为 [out, in//2] dtype uint8，并添加 MXFP4_PACK_FACTOR = 2 常量。
2. 在 linear_method_npu.py 中简化 NPUSingleLevelMXFP4OfflineLinearMethod 的 process_weights_after_loading 方法：去除通过 npu_dtype_cast 重新打包 FP4 的步骤，因为权重已经是打包格式，只需转置即可。
3. 更新两个文件的文档字符串，清楚说明新格式为 packed FP4。
4. 保留原有的 matmul 调用（npu_quant_matmul）和缩放因子处理逻辑不变。

关键文件：

- python/sglang/srt/layers/quantization/modelslim/schemes/modelslim_mxfp4.py（模块 量化方案；类别 source；类型 data-contract；符号 MXFP4_PACK_FACTOR, ModelSlimMXFP4Scheme.create_weights）：核心数据契约变更：调整 weight 参数形状和 dtype 以匹配新的 packed 格式。
- python/sglang/srt/hardware_backend/npu/quantization/linear_method_npu.py（模块 NPU 量化；类别 source；类型 core-logic；符号 NPUSingleLevelMXFP4OfflineLinearMethod.process_weights_after_loading）：核心后处理逻辑简化：去除冗余的 dtype_cast 重新打包，直接转置已打包权重。

关键符号：ModelSlimMXFP4Scheme.create_weights, NPUSingleLevelMXFP4OfflineLinearMethod.process_weights_after_loading

关键源码片段

python/sglang/srt/layers/quantization/modelslim/schemes/modelslim_mxfp4.py

核心数据契约变更：调整 weight 参数形状和 dtype 以匹配新的 packed 格式。

```
# python/sglang/srt/layers/quantization/modelslim/schemes/modelslim_mxfp4.py

# 常量定义：msmodelslim 导出的 W4A4_MXFP4 格式固定 group_size=32，打包因子为 2。
MXFP4_BLOCK_SIZE = 32
MXFP4_PACK_FACTOR = 2 # 每个 uint8 包含两个 FP4 值

class ModelSlimMXFP4Scheme(ModelSlimLinearScheme):
    """W4A4_MXFP4 offline scheme — packed-FP4 weights, MXFP4 activations."""

    def create_weights(self, layer, input_size_per_partition, output_partition_sizes,
                       input_size, output_size, params_dtype, **extra_weight_attrs):
        weight_loader = extra_weight_attrs.get('weight_loader')
        output_size_per_partition = sum(output_partition_sizes)

        # msmodelslim 将两个 FP4 值打包在一个 uint8 中，沿着输入维度折叠。
        weight = ModelWeightParameter(
            data=torch.empty(
                (output_size_per_partition, input_size_per_partition // MXFP4_PACK_FACTOR),
                dtype=torch.uint8,
            ),
            input_dim=1,
            output_dim=0,
            weight_loader=weight_loader,
        )
        layer.register_parameter('weight', weight)

        # 缩放因子仍然是 uint8，形状 [out, in//32]
        scale_dim = input_size_per_partition // MXFP4_BLOCK_SIZE
        weight_scale = GroupQuantScaleParameter(
            data=torch.empty((output_size_per_partition, scale_dim), dtype=torch.uint8),
            input_dim=1,
            output_dim=0,
            weight_loader=weight_loader,
        )
        layer.register_parameter('weight_scale', weight_scale)

    def process_weights_after_loading(self, layer):
        # 委托给 kernel 的 process_weights_after_loading，它处理转置和缩放重塑。
        self.kernel.process_weights_after_loading(layer)

    def apply_weights(self, layer, x, bias=None):
        return self.kernel.apply(layer, x, bias)
```

python/sglang/srt/hardware_backend/npu/quantization/linear_method_npu.py

核心后处理逻辑简化：去除冗余的 dtype_cast 重新打包，直接转置已打包权重。

```
# python/sglang/srt/hardware_backend/npu/quantization/linear_method_npu.py

class NPUSingleLevelMXFP4OfflineLinearMethod(NPUSingleLevelMXFP4LinearMethod):
    """Ascend NPU offline W4A4 (ModelSlim W4A4_MXFP4): packed FP4 weights."""

    def process_weights_after_loading(self, layer: torch.nn.Module) -> None:
        weight = layer.weight.data
        if not weight.is_npu:
            weight = weight.to(f'npu:{torch.npu.current_device()}')
        # 权重已经是 packed uint8 [out, in//2]，直接转置为 [in//2, out]
        # 保持 strided view，使得缩放映射正确。
        layer.weight = Parameter(weight.transpose(0, 1), requires_grad=False)

        weight_scale = layer.weight_scale.data
        if not weight_scale.is_npu:
            weight_scale = weight_scale.to(f'npu:{torch.npu.current_device()}')
        # npu_quant_matmul 要求 x2Scale 是 3D: [out, in/32] -> [out, in/64, 2] -> transpose to [in/
        64, out, 2]
        n_dim, k_dim = weight_scale.shape
        layer.weight_scale = Parameter(
            weight_scale.reshape(n_dim, k_dim // 2, 2).transpose(0, 1),
            requires_grad=False,
        )
```

评论区精华

LinyuanLi0046 询问为何不在离线路径中使用 NZ 格式（FRACTAL_NZ），作者 TallMessiWu 回应称测试后未发现性能差异，因此暂不更改。

- Why not use NZ format in offline path? (design): 暂不采用 NZ 格式，当前 transpose 方案足够。

风险与影响

- 风险：数据契约变更：该 PR 改变了 ModelSlim MXFP4 权重的加载格式，使用旧格式（fp8-container）的检查点需重新导出才能加载。风险仅限于 Ascend NPU 后端，不影响其他硬件或量化方案。没有性能回归，且加载逻辑更简洁。
- 影响：影响范围仅限使用 Ascend NPU 且采用 ModelSlim W4A4_MXFP4 量化的用户。这些用户需要升级到包含此修复的 SGLang 版本，并确认其检查点已使用最新的 ModelSlim 导出工具生成。无其他影响。
- 风险标记：数据契约变更，NPU 专有路径，向后兼容风险

关联脉络

- PR #23795 :sparkles: [llm][npu][quant] Add W4A4 MXFP4 quantization support for Qwen3 Dense on Ascend NPU: Follow-up fix for initial W4A4 support, correcting the weight format assumption.
- PR #21584 [RFC][NPU] Ascend NPU A5 Support for MXFP8/MXFP4 Quantization: Related RFC for MXFP quantization on NPU, providing context for this and previous PRs.