PR #22338 完整报告

sgl-project/sglang

:sparkles: [diffusion][npu][quant] Add MXFP4 quantization support for Wan2.2 Diffusion on Ascend NPU

合并时间：2026-05-19 12:46

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/22338

执行摘要

- 一句话：为 Wan2.2 Diffusion 添加 Ascend NPU MXFP4 量化支持
- 推荐动作：建议精读该 PR，它展示了如何在 diffusion 子系统中集成新的量化方法，代码模块化良好（在线 vs 离线分离），且 review 中涉及的性能优化和设计决策值得借鉴。需特别关注在线量化路径的实验性标注以及加载器逻辑对 modelslim 显式标志的改动。

功能与动机

作为 NPU 量化路线图（#14424）的一部分，继 MXFP8 支持（#20922）之后，本 PR 为 Wan2.2 扩散模型添加 MXFP4 量化支持，以在 Ascend NPU A5 系列上进一步降低显存占用并提升推理性能。PR body 详细描述了在线量化和离线预量化两种路径的动机。

实现拆解

1. 新增在线量化配置：创建 mxfp4_npu.py，定义 NPUMXFP4Config（注册为 mxfp4_npu）和 NPUMXFP4DiffusionLinearMethod。在模型加载时，create_weights 以 FP16/BF16 创建权重，process_weights_after_loading 调用 npu_dynamic_dual_level_mx_quant 将权重量化到 MXFP4，并转为 FRACTAL_NZ 格式。前向时，apply_weights 对激活进行量化并执行 npu_dual_level_quant_matmul。
2. 新增离线预量化加载方案：创建 modelslim_mxfp4_scheme.py，实现 ModelSlimMXFP4Scheme。create_weights 根据 msmodelslim 导出的 tensor 格式（float8_e4m3fn 权重、uint8 L1 scale、float32 L0 dual scale、float32 mul_scale）注册参数。process_weights_after_loading 将权重转为 float4_e2m1fn_x2 和 FRACTAL_NZ 格式，并将 dual scale 转置为 API 所需形状。apply_weights 在前向时应用 mul_scale（仅在非全 1 时），然后进行双级量化矩阵乘法。
3. 注册与分派：更新 __init__.py，将 NPUMXFP4Config 注册到 QUANTIZATION_METHODS 字典；在 modelslim.py 的 _get_scheme_from_parts 中添加 W4A4_MXFP4 / W4A4_MXFP4_DUALSCALE 分支，分派到 ModelSlimMXFP4Scheme。
4. 加载器与配置增强：修改 transformer_load_utils.py 中的 _resolve_quant_config，当显式指定 --quantization modelslim 时从组件目录加载量化描述文件，而非返回空配置；同时增加 safetensors 元数据回退路径。
5. 权重重命名工具升级：更新 wan_repack.py，添加 .linear. 和 .div. 到 RENAME_DICT 以匹配 MXFP4 msmodelslim 输出；支持多 shard safetensors 加载。

6. 辅助修改：更新 server_args.py 帮助文本澄清 mxfp4_npu 与 mxfp4 的目标平台区别；修复 fp8.py 中 AMD AITER 路径的权重复制方式。

关键文件：

- python/sglang/multimodal_gen/runtime/layers/quantization/mxfp4_npu.py（模块 量化层；类别 source；类型 core-logic；符号 NPUMXFP4Config, init, get_name, get_supported_act_dtypes）：核心新增文件：定义在线 MXFP4 量化配置（NPUMXFP4Config）和线性方法（NPUMXFP4DiffusionLinearMethod），包含权重创建、在线量化、前向调用等关键逻辑。
- python/sglang/multimodal_gen/runtime/layers/quantization/modelslim_mxfp4_scheme.py（模块 量化层；类别 source；类型 data-contract；符号 ModelSlimMXFP4Scheme, create_weights, process_weights_after_loading, apply_weights）：核心新增文件：实现 ModelSlimMXFP4Scheme，定义离线预量化权重的加载方案，包括权重、双级 scale 和 mul_scale 的参数注册与前处理。
- python/sglang/multimodal_gen/runtime/layers/quantization/modelslim.py（模块 量化层；类别 source；类型 data-contract）：修改 ModelSlimConfig._get_scheme_from_parts，新增 W4A4_MXFP4/W4A4_MXFP4_DUALSCALE 分支，分派到 ModelSlimMXFP4Scheme，并增强 NotImplementedError 信息。
- python/sglang/multimodal_gen/runtime/loader/transformer_load_utils.py（模块 模型加载；类别 source；类型 core-logic）：修改 _resolve_quant_config：当显式 --quantization modelslim 时加载量化描述文件，并增加 safetensors 元数据回退路径。
- python/sglang/multimodal_gen/runtime/layers/quantization/__init__.py（模块 量化层；类别 source；类型 dependency-wiring）：注册 NPUMXFP4Config 到 QUANTIZATION_METHODS, 添加 mxfp4_npu 到 QuantizationMethods literal。
- python/sglang/multimodal_gen/tools/wan_repack.py（模块 工具脚本；类别 source；类型 core-logic）：添加 .linear. 和 .div. 到 RENAME_DICT 以支持 MXFP4 msmodelslim 权重重命名；支持多 shard safetensors 加载。

关键符号：create_weights, process_weights_after_loading, apply_weights, get_quant_method, _get_scheme_from_parts, _resolve_quant_config, load_sharded_safetensors

关键源码片段

python/sglang/multimodal_gen/runtime/layers/quantization/mxfp4_npu.py

核心新增文件：定义在线 MXFP4 量化配置（NPUMXFP4Config）和线性方法（NPUMXFP4DiffusionLinearMethod），包含权重创建、在线量化、前向调用等关键逻辑。

```
class NPUMXFP4Config(QuantizationConfig):
    # 在线 MXFP4 配置类，注册为 "mxfp4_npu"
    @classmethod
    def get_name(cls) -> str:
        return "mxfp4_npu" # 与 ROCm 的 "mxfp4" 区分

    def get_quant_method(self, layer, prefix):
```

```
        if isinstance(layer, LinearBase):
            return NPUMXFP4DiffusionLinearMethod(self)
        return None


class NPUMXFP4DiffusionLinearMethod(LinearMethodBase):
    # 在线双级量化线性方法
    def create_weights(self, layer, input_size_per_partition, output_partition_sizes,
                       input_size, output_size, params_dtype, **extra_weight_attrs):
        # 在线模式：先创建原始精度的权重，后续在 process_weights_after_loading 中量化
        output_size_per_partition = sum(output_partition_sizes)
        weight_loader = extra_weight_attrs.get("weight_loader")
        layer.logical_widths = output_partition_sizes
        layer.input_size_per_partition = input_size_per_partition
        layer.output_size_per_partition = output_size_per_partition
        layer.orig_dtype = params_dtype
        weight = ModelWeightParameter(
            data=torch.empty(output_size_per_partition, input_size_per_partition,
                             dtype=params_dtype),
            input_dim=1, output_dim=0, weight_loader=weight_loader
        )
        layer.register_parameter("weight", weight)
```

python/sglang/multimodal_gen/runtime/layers/quantization/modelslim_mxfp4_scheme.py

核心新增文件：实现 ModelSlimMXFP4Scheme，定义离线预量化权重的加载方案，包括权重、双级 scale 和 mul_scale 的参数注册与前处理。

```
class ModelSlimMXFP4Scheme(ModelSlimLinearScheme):
    # 离线 MXFP4 方案：加载 msmodelslim 预量化权重
    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)

        # 权重：msmodelslim 导出为 float8_e4m3fn，后续在 process_weights_after_loading 中转为 float4_e2m1fn_x2
        weight = ModelWeightParameter(
            data=torch.empty((output_size_per_partition, input_size_per_partition),
                             dtype=torch.float8_e4m3fn),
            input_dim=1, output_dim=0, weight_loader=weight_loader
        )
        layer.register_parameter("weight", weight)

        # L1 块尺度：uint8 [out, in/32]，e8m0 编码 +127 偏移
        scale_dim = input_size_per_partition // MXFP4_BLOCK_SIZE # 32
        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)

    # L0 粗尺度：float32 [out, in/512, 1]，每个 L0 块覆盖 16 个 L1 块 (512 元素)
    dual_scale_dim = scale_dim // MXFP4_DUAL_LEVEL_RATIO # 16
    weight_dual_scale = GroupQuantScaleParameter(
        data=torch.empty((output_size_per_partition, dual_scale_dim, 1), dtype=torch.float32),
        input_dim=1, output_dim=0, weight_loader=weight_loader
    )
    layer.register_parameter("weight_dual_scale", weight_dual_scale)

    # Smooth Quant 激活尺度 (mul_scale)：必须应用以避免摩尔纹
    mul_scale = BasevLLMParameter(
        data=torch.empty((input_size_per_partition,), dtype=torch.float32),
        weight_loader=weight_loader
    )
    mul_scale.missing_param_init = "ones" # 若 checkpoint 中缺失则初始化为全 1（无操作）
    layer.register_parameter("mul_scale", mul_scale)
```

评论区精华

1. 性能优化：gemini-code-assist 指出前向中 torch.all(mul_scale == 1.0) 导致 GPU-CPU 同步，建议将检查移至权重加载阶段并预存布尔标志。该建议已被采纳。
2. 设计讨论：OrangeRedeng 询问为何创建独立的 mxfp4_npu 量化实体而非将其纳入 modelslim。TallMessiWu 解释在线量化与离线预量化机制不同，前者在加载时实时量化 FP16 权重，后者仅加载已量化权重，因此需要独立配置。OrangeRedeng 表示理解。
3. 代码风格：gemini-code-assist 建议将 fp8.py 中直接赋值 .data 的方式改为 torch.no_grad() + copy_() 以保持 Parameter 身份。最终提交已采用该模式。

- mul_scale == 1.0 检查导致 GPU-CPU 同步 (performance)：已被采纳，代码中预先计算 use_mul_scale 布尔标志。
- 为何创建独立的 mxfp4_npu 实体而非整合到 modelslim (design)：OrangeRedeng 理解设计理由，线程解决。
- fp8.py 中 .data 赋值改为 copy_ (correctness)：已在最终提交中采用 copy_ 模式。

风险与影响

- 风险：在线量化路径（npu_dynamic_dual_level_mx_quant）是实验性的，可能产生与离线校准路径不同的数值结果，需要在关键应用场景中进行验证。离线路径依赖于 msmodelslim 特定的 tensor 格式，若 msmodelslim 更新格式，可能需要同步修改。NPU 特有 API（torch_npu）在非 NPU 环境下不可用，需通过 _is_npu 守卫导入，防止 CI 失败。缺少直接针对 MXFP4 量化路径的测试配套，回归风险较高。
- 影响：用户可通过 --quantization mxfp4_npu 或 --quantization modelslim（配合预量化权重）启用 MXFP4 量化，在 Ascend NPU A5 系列上获得 11-14% 的 E2E 延迟提升。团队需要维护新增的 NPU 专用配置类，并与 ROCm 的 mxfp4 保持命名区分。加载器逻辑的变更（显式 modelslim 标志的处理）可能影响其他量化方式的解析，但已通过优先级注释明确。

- 风险标记：在线量化路径实验性，依赖 NPU 特有 API，离线权重格式依赖 msmodelslim，缺少直接测试配套，npu import 条件守卫

关联脉络

- PR #20922 [NPU] Add MXFP8 quantization support for Wan2.2 Diffusion on Ascend NPU: 前期 MXFP8 支持，本 PR 是 MXFP4 的延续，共享相同的架构模式。
- PR #24816 [ROCm] Add Mxfp4Config for AMD MI350+: 与本 PR 的 NPU MXFP4 配置发生命名冲突，导致注册键改为 mxfp4_npu 以共存。
- PR #14424 [NPU] [Roadmap] NPU quantization 2026 Q2 Roadmap: NPU 量化路线图 Issue，明确 MXFP8/MXFP4 支持为规划任务。