PR #20922 完整报告

sgl-project/sglang

:sparkles: [diffusion][npu][quant] Add MXFP8 quantization support for Wan2.2 Diffusion on Ascend NPU

合并时间：2026-05-08 02:30

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/20922

执行摘要

- 一句话：为 Ascend NPU 扩散模型添加 MXFP8 在线 / 离线量化支持
- 推荐动作：建议精读，特别是在线与离线方案的设计分离、NPU 专用量化层的实现，以及 wan_repack.py 的 bug 修复方法。这些模式可用于在其他硬件上扩展量化支持。

功能与动机

该 PR 填补了 Issue #14424 (NPU quantization roadmap) 中 MXFP8 支持的缺口。需求是让 Wan2.2 扩散模型能在 Ascend NPU 上利用 MXFP8 低精度计算以提高推理效率，同时提供在线和离线两种灵活部署方式。硬件要求为 Ascend A5 及以上系列。

实现拆解

1. 新增在线量化方法（ mxfp8_npu.py ）：定义 MXFP8Config 类和 NPUMXFP8DiffusionLinearMethod。在 *create_weights* 中分配 FP16/BF16 原始权重；在 process_weights_after_loading 中将权重移至 NPU 并通过 npu_dynamic_mx_quant 在线量化为 MXFP8，生成 weight_scale_inv 参数；在 *apply* 中对激活值做动态 MXFP8 量化并调用 npu_quant_matmul 完成计算。
2. 新增离线量化方案（ modelslim_mxfp8_scheme.py ）：定义 *ModelSlimMXFP8Scheme*，继承自 *ModelSlimLinearScheme*。加载 msmodelslim 预量化的 float8_e4m3fn 权重和 uint8 scale，后处理仅重塑 scale 形状，推理时激活量化 + 矩阵乘，无需重新量化权重。
3. 重构打包工具（ wan_repack.py ）：彻底改写了原脚本，修复四个阻塞性 bug（glob 模式被当作文字路径、缺少 *else* 分支导致 NameError、无条件更新 *quant_config* 导致 KeyError、model_type 不完整）。新工具支持 Wan2.2-T2V-A14B / I2V-A14B / TI2V-5B ，一步完成原始 Diffusers 模型拷贝 + 量化权重重命名 + config.json 恢复。
4. 集成到加载流程：在 transformer_load_utils.py 中优先使用 --quantization 显式参数；在 modelslim.py 中增加 W8A8_MXFP8 分支；在 init.py 中注册 MXFP8Config；在 server_args.py 中新增 --quantization 参数；在 quantization_utils.py 中放宽 glob 匹配；在 fsdp_load.py 中将 weight_scale 加入 FSDP 忽略键列表。
5. LLM 侧小幅重构（ fp8.py ）：移除 apply_fp8_marlin_linear 的直接导入，改为 torch.ops.sglang.apply_fp8_marlin_linear；调整 MOE 后处理中权重 shuffle 的写法。

6. 文档更新：在 ascend_npu_quantization.md 中新增 MXFP8 章节，更新 quantization.md 加入扩散模型量化说明。
7. 测试与性能验证：修改 test_transformer_quant.py 以适应新参数。PR 附带了 Wan2.2-TI2V-5B 在 A5 上的性能对比，显示 MXFP8 下 VBench 评分无明显衰退，但端到端时延未缩短（受限于 NPU kernel 瓶颈），仅节省显存。

关键文件：

- python/sglang/multimodal_gen/runtime/layers/quantization/mxfp8_npu.py（模块 量化层；类别 source；类型 core-logic；符号 MXFP8Config, init, get_name, get_supported_act_dtypes）：在线量化核心实现，定义 MXFP8Config 和 NPUMXFP8DiffusionLinearMethod，演示 NPU 专用量化流程
- python/sglang/multimodal_gen/runtime/layers/quantization/modelslim_mxfp8_scheme.py（模块 量化方案；类别 source；类型 data-contract；符号 ModelSlimMXFP8Scheme, create_weights, process_weights_after_loading, apply_weights）：离线量化方案核心，展示预量化权重的加载和推理流程
- python/sglang/multimodal_gen/tools/wan_repack.py（模块 打包工具；类别 source；类型 dependency-wiring；符号 get_transformer_config, update_dict_, load_sharded_safetensors, convert_transformer）：工具重构，修复四个严重 bug 并支持多模型类型，实现一键 repack
- python/sglang/srt/layers/quantization/fp8.py（模块 量化框架；类别 source；类型 dependency-wiring）：LLM 侧导入清理和 MOE 后处理调整，确保与 NPU 量化方法的兼容
- python/sglang/multimodal_gen/runtime/loader/transformer_load_utils.py（模块 加载器；类别 source；类型 dependency-wiring）：加载器支持 --quantization 参数优先级，允许显式指定量化方法
- python/sglang/multimodal_gen/runtime/layers/quantization/modelslim.py（模块 方案调度；类别 source；类型 data-contract）：增加 W8A8_MXFP8 分支，将离线 MXFP8 方案接入现有的 ModelSlim 调度
- python/sglang/multimodal_gen/runtime/layers/quantization/__init__.py（模块 注册入口；类别 source；类型 dependency-wiring）：注册 MXFP8Config 并更新 QuantizationMethods 枚举，是量化方法发现的入口
- python/sglang/multimodal_gen/runtime/server_args.py（模块 参数；类别 source；类型 core-logic）：新增 --quantization CLI 参数，使用户能显式选择量化方法
- python/sglang/multimodal_gen/runtime/utils/quantization_utils.py（模块 量化工具；类别 source；类型 core-logic）：放宽 quant_model_description*.json 的 glob 匹配，支持 repack 后的文件名
- python/sglang/multimodal_gen/runtime/loader/fsdp_load.py（模块 FSDP；类别 source；类型 core-logic）：将 weight_scale 加入 FSDP 忽略键列表，防止加载离线 MXFP8 权重时崩溃
- docs/platforms/ascend/ascend_npu_quantization.md（模块 文档；类别 docs；类型 documentation）：新增 MXFP8 量化说明，告知用户硬件要求和用法

关键符号：MXFP8Config.get_name, MXFP8Config.get_quant_method, NPUMXFP8DiffusionLinearMethod.create_weights, NPUMXFP8DiffusionLinearMethod.process_weights_after_loading, NPUMXFP8DiffusionLinearMethod.apply, ModelSlimMXFP8Scheme.create_weights, ModelSlimMXFP8Scheme.process_weights_after_loading, ModelSlimMXFP8Scheme.apply_weights, convert_transformer, load_sharded_safetensors, _resolve_quant_config

关键源码片段

python/sglang/multimodal_gen/tools/wan_repack.py

工具重构，修复四个严重 bug 并支持多模型类型，实现一键 repack

```
# 关键修复：load_sharded_safetensors 使用 glob 模式正确查找文件
# 原脚本使用 pathlib.Path(dir, "*model*.safetensors") 当作文字路径，导致 FileNotFoundError

def load_sharded_safetensors(directory: pathlib.Path, pattern: str) -> dict:
    candidates = sorted(directory.glob(pattern))
    if not candidates:
        raise FileNotFoundError(f"No file matching '{pattern}' found in {directory}")
    if len(candidates) > 1:
        raise FileNotFoundError(
            f"Multiple files matching '{pattern}' found in {directory}: {candidates}"
        )
    state_dict = {}
    state_dict.update(load_file(candidates[0]))
    return state_dict

# 关键修复：convert_transformer 现在使用正确的 glob 模式并处理 quant_config
# 原脚本无条件更新 quant_config 导致 KeyError，现已改为仅对存在的键进行替换

def convert_transformer(
    model_type: str, model_dir: pathlib.Path, output_dir: pathlib.Path
) -> None:
    """将单个量化 transformer 目录转为 Diffusers 格式"""
    model_path = pathlib.Path(model_dir)
    out_path = pathlib.Path(output_dir)
    out_path.mkdir(parents=True, exist_ok=True)
    RENAME_DICT = get_transformer_config(model_type)

    # 使用 glob 模式加载 safetensors
    state_dict = load_sharded_safetensors(model_path, "quant_model_weight*.safetensors")

    # 使用 glob 模式查找描述文件
    json_candidates = sorted(model_path.glob("quant_model_description*.json"))
    if not json_candidates:
        raise FileNotFoundError(
            f"No quant_model_description*.json found in {model_path}"
```

```
        )
    with open(json_candidates[0]) as f:
        quant_config = json.load(f)

    # 重命名键并更新 quant_config（仅对存在的键更新，避免 KeyError）
    for key in list(state_dict.keys()):
        new_key = key[:]
        for replace_key, rename_key in RENAME_DICT.items():
            new_key = new_key.replace(replace_key, rename_key)
        update_dict_(state_dict, key, new_key)
        # 仅当旧键存在于 quant_config 中才替换
        if key in quant_config:
            update_dict_(quant_config, key, new_key)
    # ... 后续保存
```

评论区精华

- 硬件兼容性：iforgetmyname 质疑 NPU 是否真的支持 FP8，OrangeRedeng 澄清 “A5 works with mxfp8 (and even with mxfp4)”，最终约定在文档中明确标注 A5 系列要求。
- 架构分层：TamirBaydasov 建议将 MXFP8 线性方法拆分为 fp8.py 中的 MXFP8LinearAscendMethod（定义权重）和硬件后端中的 NPUMXFP8LinearMethod（权重处理与 kernel），作者采纳并重构；后来将 LLM 侧方法分离到单独 PR，本 PR 只保留扩散路径。
- 代码风格：ping1jing2 要求使用 init_logger 代替 logging、将 import 移到顶部、添加 flatten 输入的解释注释——作者逐一修复。
- 文档补充：OrangeRedeng 建议更新 ascend_npu_quantization.md，作者完成。
- 测试与 CI：ping1jing2 要求提供准确性和性能数据并上传权重到 CI 服务器；由于 A5 CI 尚未就绪（依赖 #24540），CI 测试无法通过但经团队确认失败均与 PR 无关，最终合并。
  - MXFP8 硬件兼容性 (question): 确认仅 A5 以上支持，文档中标注要求
  - MXFP8 量化架构分层 (design): 拆分完成，LLM 部分移至后续 PR，本 PR 仅含扩散路径
  - 代码风格：使用 init_logger (style): 已改正
  - 文档更新要求 (documentation): 已完成文档更新
  - 性能与 CI 测试要求 (testing): 作者提供了性能报告；CI 失败经分析均与 PR 无关，已合并，但需等待 #24540 才能启用 A5 CI

风险与影响

- 风险：
  1. 硬件依赖风险：仅 A5 及以上支持，若在 A2/A3 调用 npu_dynamic_mx_quant 将触发运行时错误，当前未在代码中加兼容性检查或警告。
  2. 在线量化与 CPU offload 冲突：在 NPUMXFP8DiffusionLinearMethod.process_weights_after_loading 中，由于 dit_cpu_offload 默认将参数移回 CPU，代码显式将权重移至 NPU 后再量化。这虽然正确工作，但与 offload 意图矛盾，可能导致大模型显存不足。

3. 离线量化格式耦合：ModelSlimMXFP8Scheme 紧密依赖 msmodelslim 的权重排列（float8_e4m3fn 权重 + uint8 scale），若上游工具更改输出格式，加载将静默损坏。
4. LLM 侧分离：LLM MXFP8 支持被推迟，可能导致 fp8.py 中当前改动（如导入清理）与未来 LLM 量化方法冲突。
5. 测试覆盖不足：新增核心文件（mxfp8_npu.py、modelslim_mxfp8_scheme.py）缺少独立的单元测试；CI 中扩散量化测试因硬件不可用被跳过。 - 影响：对用户：提供 --quantization mxfp8 选项启用扩散模型 MXFP8 量化；使用 wan_repack.py 可转换预量化权重，减少模型加载时间和存储空间，但需注意硬件限制。 对系统：增加了约 700 行代码，引入了新的量化配置和线性方法，但不影响现有量化流程。 对团队：需维护两个新增量化方案；后续 LLM MXFP8 PR 可能带来重构。

- 风险标记：硬件依赖 A5 及以上，在线量化与 CPU offload 交互，离线量化格式依赖 msmodelslim, LLM 量化分离需等待后续 PR, 测试覆盖不足

关联脉络

- PR #14424 [NPU] [Roadmap] NPU quantization 2026 Q1 Roadmap: 本 PR 是 roadmap 中 MXFP8 支持的一部分，close 了相关 gap
- PR #24540 [CI] ...（尚未合并但关联）: 该 PR 将启用 A5 CI，使得 MXFP8 的自动化测试成为可能
- PR #17936 [diffusion] Support quantization for diffusion models: 扩散模型量化的基础 issue，本 PR 实现了其中的 MXFP8 部分