PR #30768 完整报告

sgl-project/sglang

:sparkles: [llm][npu][quant] Add W8A8 MXFP8 quantization for Qwen3 MoE on Ascend NPU

合并时间：2026-07-29 15:39

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/30768

执行摘要

- 一句话：为 NPU A5 上的 Qwen3 MoE 模型添加 W8A8 MXFP8 量化
- 推荐动作：该 PR 设计周密，值得仔细阅读，特别是以下方面：融合路由量化如何避免额外的量化 kernel 启动、在线 / 离线方法如何共享同一套 NPUMXFP8MoEMethod 内核，以及 *create_moe_runner* 中通过 *isinstance* 检查 *moe_runner_backend* 以防止错误分发。同时，审查反馈中的迭代过程（继承 vs 组合、内核提取、DeepEP 支持）展示了良好的代码演化实践，可供团队借鉴。

功能与动机

在 Ascend NPU A5 上实现 MXFP8 量化可以显著提升推理性能和内存效率。此 PR 是 issue #21584 中规划的一部分，该 issue 概述了在 NPU A5 上支持各种模型系列（扩散、LLM、VLM）的 MXFP8/MXFP4 量化。具体而言，此 PR 实现了 Qwen3 MoE 模型的 W8A8 MXFP8 支持，是之前密集模型 W8A8（PR #22352）和 W4A4（PR #23795）的延续。

实现拆解

1. 配置分发：在 Fp8Config.get_quant_method() 中为 NPU 添加 FusedMoE 分支，返回新的 NPUMXFP8OnlineMoEMethod，实现在线量化入口。
2. 在线 MoE 方法：创建 NPUMXFP8OnlineMoEMethod（继承 UnquantizedFusedMoEMethod），仅重写 *create_moe_runner*，将 NPUMXFP8MoEMethod 实例作为 w13_kernel / w2_kernel 附加到层上，其余权重创建和后处理逻辑继承自未量化方法。
3. 核心 MoE 方法：实现 NPUMXFP8MoEMethod（继承 _NPUMoEMethodBase），包含权重在线量化（*_quantize_weight_online* 使用 npu_dynamic_mx_quant 对 3D 权重进行量化）、权重后处理（转换为 FRACTAL_NZ 布局 via npu_format_cast）以及前向传播。前向中，gmm1 使用 GroupedMatmulSwigluQuant（融合 gate+up+swiglu+requant，返回量化后的激活和块缩放），gmm2 使用 GroupedMatmul。
4. 离线 MoE Scheme：创建 ModelSlimMXFP8MoEScheme（继承 ModelSlimMoEScheme），加载预量化的 float8_e4m3fn 权重和 uint8 块缩放，并在 process_weights_after_loading 中委托给 NPUMXFP8MoEMethod 进行布局转换。
5. 融合路由量化：在 npu_moe_init_routing_v2 中通过 *quant_mode=3* 在路由时一同完成激活量化，避免额外调用 npu_dynamic_mx_quant。通过 _normalize_mxfp_scale 将扁平缩

放因子转换为 pair-split 布局以匹配后续内核需求。

6. 路由门量化修复：离线检查点中 mlp.gate 的权重量化现在通过 quant_model_description.json 描述驱动，而不是硬编码为 BF16，从而正确应用 MXFP8 缩放因子。
7. 动态量化封装：将 hidden_states_quant.py 重命名为 quant.py，并让 HiddenStatesDynamicQuant 根据 dtype 选择 npu_dynamic_mx_quant（MXFP8）或 npu_dynamic_quant（INT8/INT4），使模块可复用。
8. 文档：更新了 ascend_npu_quantization.mdx 和 quantization.mdx，描述 MXFP8 MoE 的使用方法。
9. 测试：PR 未包含直接对应的测试文件，仅通过端到端启动脚本验证。

关键文件：

- python/sglang/srt/layers/quantization/modelslim/schemes/modelslim_mxfp8_moe.py（模块 量化方案；类别 source；类型 data-contract；符号 ModelSlimMXFP8MoEScheme, init, create_weights, process_weights_after_loading）：新增文件，实现离线 MXFP8 MoE scheme，定义权重创建和布局转换委托。
- python/sglang/srt/hardware_backend/npu/quantization/moe_methods.py（模块 MoE 方法；类别 source；类型 dependency-wiring；符号 _require_e8m0_dtype, NPUMXFP8MoEMethod, init, _quantize_weight_online）：核心文件，实现 NPUMXFP8MoEMethod，包含在线权重量化、前向传播以及量化和非量化权重的分支处理。
- python/sglang/srt/hardware_backend/npu/moe/matmul.py（模块 内核封装；类别 source；类型 dependency-wiring；符号 GroupedMatmulSwigluQuant, forward）：新增 GroupedMatmulSwigluQuant 类，封装 fused gate+up+swiglu+requant 内核，供 MXFP8 及未来 block-scaled 方案复用。

关键符号：ModelSlimMXFP8MoEScheme.init, ModelSlimMXFP8MoEScheme.create_weights, ModelSlimMXFP8MoEScheme.process_weights_after_loading, NPUMXFP8MoEMethod.init, NPUMXFP8MoEMethod._quantize_weight_online, NPUMXFP8MoEMethod.process_weights_after_loading, NPUMXFP8MoEMethod.apply, NPUMXFP8MoEMethod.apply_fused_gmm1_swiglu, NPUMXFP8OnlineMoEMethod.init, NPUMXFP8OnlineMoEMethod.create_moe_runner, GroupedMatmulSwigluQuant.forward, _normalize_mxfp_scale, HiddenStatesDynamicQuant.init

关键源码片段

python/sglang/srt/hardware_backend/npu/moe/matmul.py

新增 GroupedMatmulSwigluQuant 类，封装 fused gate+up+swiglu+requant 内核，供 MXFP8 及未来 block-scaled 方案复用。

```
from abc import ABC, abstractmethod
from typing import Tuple

import torch
```

```
class BaseMatmul(ABC):
    @abstractmethod
    def forward(self, ...):
        pass


class GroupedMatmul(BaseMatmul):
    # ... 已有实现 ...


class GroupedMatmulSwigluQuant(BaseMatmul):
    """Grouped matmul with swiglu and requantisation fused into one kernel.

    Used for the gate/up projection (gmm1) of block-scaled MoE: the kernel
    returns (quantized_activations, block_scale) instead of a single tensor.
    Unlike ``GroupedMatmul`` it takes no ``output_dtype`` — output dtype comes
    from ``quant_dtype`` in ``scale_args``.
    """

    def forward(
        self,
        layer: torch.nn.Module,
        weight_prefix: str,
        hidden_states: torch.Tensor,
        expert_tokens: torch.Tensor,
        output_dtype: torch.dtype = None,
        group_list_type: int = 1,
        transposed: bool = True,
        **scale_args,
    ) -> Tuple[torch.Tensor, torch.Tensor]:
        weight = getattr(layer, f"{weight_prefix}_weight", None)
        if weight is None:
            raise AttributeError(
                f"Weight attribute '{weight_prefix}_weight' not found in layer"
            )
        # This op requires a CUMULATIVE group_list; the dispatcher produces
        # COUNT form (group_list_type=1). Convert here.
        group_list = (
            expert_tokens.cumsum(0) if group_list_type == 1 else expert_tokens
        )
        return torch.ops.npu.npu_grouped_matmul_swiglu_quant_v2(
            x=hidden_states,
            weight=[weight] if transposed else [weight.transpose(1, 2)],
            group_list=group_list,
            **scale_args,
        )
```

评论区精华

审查中主要讨论了以下设计决策：

- DeepEP 支持：OrangeRedeng 指出 DeepEP 即使不在分发器中启用量化也能与 MXFP8 MoE 工作，应避免硬拒绝。TallMessiWu 接受了建议，移除了拒绝逻辑，并让方法在 DeepEP 路径下自动量化激活。
- 提取 GroupedMatmulSwigluQuant：OrangeRedeng 要求将融合内核封装到 matmul.py 中以便其他量化方案复用。TallMessiWu 将其提取为独立类，并保持与其他 GroupedMatmul 一致的接口。
- 继承未量化方法：OrangeRedeng 建议在线入口点继承 UnquantizedFusedMoEMethod 以避免重复 create_weights/process_weights/apply 等代码。TallMessiWu 重构为 NPUMXFP8OnlineMoEMethod，移除了约 85 行重复代码。
- FRACTAL_NZ 布局：OrangeRedeng 询问是否尝试 npu_format_cast，TallMessiWu 测试后确认其带来 1.4% 解码和 3.8% 预填充性能提升，并合入。
- 离线标志文档：OrangeRedeng 指出 --quantization modelslim 对离线检查点不是必需的（描述文件自动加载），TallMessiWu 修复了文档示例。
- 动态量化封装：OrangeRedeng建议将hidden_states_quant.py改造为可复用内核包装器。TallMessiWu 重命名为 quant.py 并根据 dtype 路由算子。
  - DeepEP 与 MXFP8 MoE 的兼容性 (design): TallMessiWu 移除了硬拒绝，使 DeepEP 路径正常工作：在 process_weights_after_loading 中根据后端设置输出 dtype，并在 apply_fused_gmm1_swiglu 中当 pertoken_scale 为 None 时自行量化激活。
  - 将 GroupedMatmulSwigluQuant 提取到 matmul.py (design): TallMessiWu 将 GroupedMatmulSwigluQuant 提取到 matmul.py，与其他 GroupedMatmul 保持相同风格，并添加了 group_list 累计转换。
  - 继承 UnquantizedFusedMoEMethod 避免代码重复 (design): TallMessiWu 重构为 NPUMXFP8OnlineMoEMethod(UnquantizedFusedMoEMethod)，仅重写 create_moe_runner，删除约 85 行重复代码。
  - FRACTAL_NZ 布局性能优化 (performance): TallMessiWu 测试后确认对 MXFP8 权重使用 NZ 布局带来 1.4% 解码和 3.8% 预填充性能提升，已合入。
  - 离线模式文档标志不必要 (documentation): TallMessiWu 更新了文档，移除了多余的标志，并补充说明 flagless 加载行为。
  - 动态量化内核封装 (design): TallMessiWu 将文件重命名为 quant.py，并在 HiddenStatesDynamicQuant 中根据 dtype 选择 npu_dynamic_mx_quant 或 npu_dynamic_quant。

风险与影响

- 风险：
  1. 硬件依赖性：使用了 Ascend A5 特有的内核（npu_grouped_matmul_swiglu_quant_v2, npu_dynamic_mx_quant），在 A2/A3 或非 NPU 平台上会失败。
  2. 融合路由量化正确性：_normalize_mxfp_scale 的布局转换假设了特定的输出形状，如果底层算子实现变化可能导致数值错误。已在 A5 上通过对比验证，但缺乏自动化回归测试。

3. 离线路径路由门修复：dc9aed67 中修改了 qwen3_moe.py 的 SparseMoeBlock，使得路由门的量化通过描述驱动。若其他模型也存在类似固化的 BF16 转换，可能被遗漏。
4. 缺少测试覆盖：PR 未包含专门的单元测试或集成测试，仅依赖端到端基准验证。后续重构或内核升级可能引入回归。
5. 性能数据局限性：基准测试仅在单一模型和硬件规模上运行，不同配置下的加速比可能有所差异。 - 影响：用户影响：Ascend NPU A5 用户现在可以以 W8A8 MXFP8 精度运行 Qwen3-30B-A3B 等 MoE 模型，获得约 44% 的吞吐量提升和约 30% 的延迟降低，同时几乎保持无损精度。系统影响：扩展了 NPU 后端量化栈，新增了在线 / 离线 MoE 量化路径，但未影响 CUDA 或其他后端。FusedMoE 的分发逻辑在 CPU/GPU 路径上保持不变。团队影响：维护者需要关注 Ascend A5 内核的兼容性，以及融合路由量化模式在不同 torch_npu 版本下的行为。文档已同步更新。

- 风险标记：仅支持 A5 硬件，新融合内核依赖，缺少测试覆盖，路由门量化修复影响面

关联脉络

- PR #21584 [RFC][NPU] Ascend NPU A5 Support for MXFP8/MXFP4 Quantization: 总体跟踪 issue，本 PR 是其 MoE 部分的具体实现。
- PR #22352 :sparkles: [llm][npu][quant] Add W8A8 MXFP8 quantization support for Qwen3 Dense on Ascend NPU: 密集模型 W8A8 支持，本 PR 复用了其基础设施（Fp8Config 分发、NPUMXFP8LinearMethod、ModelSlimMXFP8Scheme）。
- PR #23795 :sparkles: [llm][npu][quant] Add W4A4 MXFP4 quantization support for Qwen3 Dense on Ascend NPU: 密集模型 W4A4 支持，与本 PR 同属 NPU 量化系列，共享底层 utils。
- PR #25663 [Refactor] Unified Ascend MoE execution stack (AscendRunnerCore + dispatcher + moe_methods): 上游重构 PR，本 PR 在合并时进行了适配，将原有 fused_moe_method_npu.py 中的逻辑移植到新的 moe_methods / runner 结构中。