PR #27204 完整报告

sgl-project/sglang

[AMD] Implement QuarkW4A8MXFp4MoE to support amd/gpt-oss-120b-w-mxfp4-a-fp8

合并时间：2026-06-30 16:24

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27204

执行摘要

- 一句话：实现对 Quark W4A8 MXFP4-FP8 量化方案的 AMD 支持
- 推荐动作：此 PR 值得精读，特别是设计师处理不同量化检查点布局时的思路。关键决策包括：将 per-expert 加载器分离到独立文件、使用环境变量控制内核布局、在自动检测中添加新方案而不破坏现有逻辑。偏好精度测试应尽快补充非 nightly 的冒烟测试以提高 CI 覆盖。

功能与动机

PR body 说明："This PR extends the Quark quantization scheme to support W4A8 MXFP4-FP8 path so SGLang can load and run AMD Quark per-expert MoE checkpoints through the AITER fused-MoE backend. The main motivation is enabling support for amd/gpt-oss-120b-w-mxfp4-a-fp8 (checkpoint carries fp8 scaling for activation - pre-calib)."

实现拆解

实现拆解如下：

1. 新增 MoE 量化方案类：在 python/sglang/srt/layers/quantization/quark/schemes/quark_w4a8_mxfp4_moe.py 中创建 QuarkW4A8MXFp4MoE 类，继承 QuarkMoEScheme。__init__ 方法验证权重量化配置（静态 per-group FP4，group_size=32，e8m0）和激活量化配置（静态 per-tensor fp8_e4m3/fp8_e4m3fn）。create_weights 方法分配 AITER 对齐所需的 padded 权重缓冲区，并设置 quant_method 为 BLOCK。process_weights_after_loading 和 create_moe_runner 根据环境变量 SGLANG_USE_AITER_MOE_GU_ITLV 选择 gate/up 布局（分离或交错）。apply_weights 调用 AITER 融合 MoE kernel。
2. 新增 Quark per-expert 权重加载器：在 python/sglang/srt/layers/quantization/quark/weights.py 中添加 load_gptoss_weight_quark 函数，它使用正则拆分检查点权重：per-expert 的权重、scale、bias 交给 _load_gptoss_quark_expert_weights，其余委托给 _load_normal_weights。_load_gptoss_quark_expert_weights 根据 TP/EP 分片计算切片范围，将每个专家的 gate_up_proj（分离 gate/up 后）和 down_proj 的权重 /scale 复制到 padded 参数窗口。

3. 扩展模型加载路由：在 python/sglang/srt/models/gpt_oss.py 的 load_weights 方法中增加 elif quant_config_name == "quark" 分支，导入并调用 load_gptoss_weight_quark。
4. 增强自动检测和注册：在 python/sglang/srt/layers/quantization/quark/quark.py 中添加 _is_mx_w4a8 方法，识别 W4A8 MXFP4-FP8 配置，并在 get_moe_scheme 中路由到 QuarkW4A8MXFp4MoE。同时更新 __init__.py 导出新类。
5. 添加精度测试：在 test/registered/amd/accuracy/mi35x/test_gpt_oss_w4a8_mxfp4_eval_mi35x.py 中注册 nightly AMD MI35x 测试，使用 GSM8K 200 题评估模型准确性，阈值 0.79。测试配置 TP=8、chunked-prefill-size=130172、Triton 注意力后端，并设置环境变量 SGLANG_USE_AITER=1 和 SGLANG_USE_AITER_MOE_GU_ITLV=1。

关键文件：

- python/sglang/srt/layers/quantization/quark/schemes/quark_w4a8_mxfp4_moe.py（模块 MoE 量化；类别 source；类型 dependency-wiring；符号 QuarkW4A8MXFp4MoE, init, get_min_capability, create_weights）：核心 MoE 量化方案实现，定义了权重创建、后处理、runner 创建和 apply 逻辑。
- python/sglang/srt/layers/quantization/quark/weights.py（模块 权重加载；类别 source；类型 core-logic；符号 load_gptoss_weight_quark, _load_gptoss_quark_expert_weights）：实现 Quark per-expert 权重加载器，将检查点的分离权重 /scale 填充到 AITER 所需的 padded 缓冲区。
- test/registered/amd/accuracy/mi35x/test_gpt_oss_w4a8_mxfp4_eval_mi35x.py（模块 精度测试；类别 test；类型 test-coverage；符号 ModelConfig, post_init, get_one_example, get_few_shot_examples）：AMD MI35x 上 GPT-OSS W4A8 MXFP4-FP8 模型的 GSM8K 精度测试，注册为 nightly 套件。
- python/sglang/srt/layers/quantization/quark/quark.py（模块 量化配置；类别 source；类型 core-logic；符号 _is_mx_w4a8）：添加 _is_mx_w4a8 检测方法和导入 QuarkW4A8MXFp4MoE，并在 get_moe_scheme 中路由。
- python/sglang/srt/models/gpt_oss.py（模块 模型入口；类别 source；类型 data-contract）：在 load_weights 中添加 Quark 分支，路由到新的权重加载器。
- python/sglang/srt/layers/quantization/quark/schemes/__init__.py（模块 方案注册；类别 source；类型 dependency-wiring）：导出新类 QuarkW4A8MXFp4MoE。

关键符号：load_gptoss_weight_quark, _load_gptoss_quark_expert_weights, _is_mx_w4a8, create_weights, process_weights_after_loading, create_moe_runner, apply_weights, get_moe_scheme

关键源码片段

python/sglang/srt/layers/quantization/quark/schemes/quark_w4a8_mxfp4_moe.py

核心 MoE 量化方案实现，定义了权重创建、后处理、runner 创建和 apply 逻辑。

```
# SPDX-License-Identifier: Apache-2.0

from __future__ import annotations
```

```
import logging
from dataclasses import replace
from typing import TYPE_CHECKING, Any

import torch

from sglang.srt.environ import envs
from sglang.srt.layers.moe import MoeRunner, MoeRunnerBackend, MoeRunnerConfig
from sglang.srt.layers.moe.utils import get_moe_weight_sizes
from sglang.srt.layers.quantization.quark.schemes import QuarkMoEScheme
from sglang.srt.layers.quantization.utils import all_close_1d
from sglang.srt.utils import (
    get_bool_env_var,
    is_gfx95_supported,
    is_hip,
    round_up,
    set_weight_attrs,
)

if TYPE_CHECKING:
    from sglang.srt.layers.moe.token_dispatcher import (
        CombineInput,
        StandardDispatchOutput,
    )

logger = logging.getLogger(__name__)

_is_shuffle_moe_mxfp4 = is_gfx95_supported()

__all__ = ["QuarkW4A8MXFp4MoE"]

_is_hip = is_hip()
_use_aiter = get_bool_env_var("SGLANG_USE_AITER") and _is_hip
if _use_aiter:
    from aiter.ops.shuffle import (
        shuffle_scale,
        shuffle_scale_a16w4,
        shuffle_weight,
        shuffle_weight_a16w4,
    )

OCP_MX_BLOCK_SIZE = 32


class QuarkW4A8MXFp4MoE(QuarkMoEScheme):
    '''Quark MoE scheme for MXFP4 weights with static FP8 activations.'''

    def __init__(self, weight_config: dict[str, Any], input_config: dict[str, Any]):
        # 验证权重配置：静态 per-group FP4，group_size=32，scale_format=e8m0
```

```
        self.weight_quant = weight_config
        self.input_quant = input_config

        weight_qscheme = self.weight_quant.get('qscheme')
        input_qscheme = self.input_quant.get('qscheme')
        weight_dtype = self.weight_quant.get('dtype')
        input_dtype = self.input_quant.get('dtype')

        if not (
            weight_dtype == 'fp4'
            and weight_qscheme == 'per_group'
            and self.weight_quant.get('group_size') == OCP_MX_BLOCK_SIZE
            and not self.weight_quant.get('is_dynamic')
            and self.weight_quant.get('scale_format') == 'e8m0'
        ):
            raise ValueError(
                'For W4A8 MXFP4-FP8 Fused MoE layers, weights must be '
                'static per-group FP4 with group_size=32 and e8m0 scales. '
                f'Found {self.weight_quant}.'
            )

        # 验证激活配置：静态 per-tensor fp8_e4m3/fp8_e4m3fn
        if not (
            input_dtype in ('fp8_e4m3', 'fp8_e4m3fn')
            and input_qscheme == 'per_tensor'
            and not self.input_quant.get('is_dynamic')
        ):
            raise ValueError(
                'For W4A8 MXFP4-FP8 Fused MoE layers, activations must be '
                'static per-tensor fp8_e4m3/fp8_e4m3fn. '
                f'Found {self.input_quant}.'
            )

        self.with_bias = False

    @classmethod
    def get_min_capability(cls) -> int:
        return 70
```

python/sglang/srt/layers/quantization/quark/weights.py

实现 Quark per-expert 权重加载器，将检查点的分离权重 /scale 填充到 AITER 所需的 padded 缓冲区。

```
import math
import re

import torch

from sglang.srt.distributed import (
```

```
    get_moe_expert_parallel_rank,
    get_moe_expert_parallel_world_size,
    get_moe_tensor_parallel_rank,
    get_moe_tensor_parallel_world_size,
)
from sglang.srt.utils import is_cuda

_is_cuda = is_cuda()


def load_gptoss_weight_quark(
    model,
    weights,
    *,
    is_nextn: bool,
    weight_name_mapping,
) -> None:
    '''加载 GPT-OSS Quark 格式的权重，分离 per-expert MoE 权重和普通权重。'''
    # 正则匹配 Quark 检查点的 per-expert 权重名称格式
    quark_expert_pat = re.compile(
        r'^(.*\.mlp\.experts)\.(\d+)\.(gate_up_proj|down_proj)\.'
        r'(weight|weight_scale|input_scale|bias)$'
    )
    quark_experts_weights = []
    normal_weights = []

    for name, weight in weights:
        if quark_expert_pat.match(name) is not None:
            quark_experts_weights.append((name, weight))
        else:
            normal_weights.append((name, weight))

    # 加载 per-expert MoE 权重到 padded 缓冲区
    quark_loaded = _load_gptoss_quark_expert_weights(
        model, quark_experts_weights, quark_expert_pat
    )
    # 剩余普通权重按默认方式加载
    model._load_normal_weights(
        normal_weights,
        is_nextn=is_nextn,
        weight_name_mapping=weight_name_mapping,
        other_loaded_param_names=quark_loaded,
    )
```

评论区精华

Review 中关键讨论：

- _is_mx_w4a8 必要性：BowenBao 询问该函数是否未使用，作者回复 "Good catch! :)" 但最终保留并用于自动检测（已解决）。
- 非 AITER 路径处理：BowenBao 建议对非 AITER 后端抛出 NotImplementedError 而非静默失败，作者添加了显式错误（已解决）。
- 权重加载器位置：HaiShaw 建议将 Quark 权重加载器移到单独的 weights.py 文件，并重命名函数以表明是 GPT-OSS 专用。作者在后续提交中进行了重构（已解决）。
- 精度测试要求：kkHuang-amd 要求提供测试命令和结果，作者在 PR 描述中补充了详细的准确性（GSM8K 0.845）和性能数据（已解决）。
- gate/up 布局检测：BowenBao 指出 is_quark_w4a8_mxfp4_format 可以泛化为 is_quark，作者回应将在依赖 PR #27201 合并后消除该逻辑（待解决 / 已规划）。
  - _is_mx_w4a8 是否未使用 (design): 函数被保留并用于自动检测，逻辑无误。
  - 非 AITER 路径的错误处理 (design): 已添加显式错误提示。
  - 权重加载器文件位置重构 (design): 已创建 quantization/quark/weights.py 并重命名函数。
  - 精度测试要求 (testing): PR body 已包含 GSM8K 准确性（0.845）和性能数据。
  - gate/up 布局自动检测泛化 (design): 暂时保留，待后续清理。

风险与影响

- 风险：技术风险：
- 非 AITER 后端限制：当前非 AITER 路径直接抛出 NotImplementedError，用户必须设置 SGLANG_USE_AITER=1，否则无法使用。
- 环境变量依赖：SGLANG_USE_AITER_MOE_GU_ITLV 控制 gate/up 布局，默认值可能与其他 AITER MoE 调用者冲突（需显式设置）。
- 硬件兼容性：测试仅在 AMD MI35x 上进行，未验证其他 AMD GPU 或 CUDA 设备。PR 中的 get_min_capability 返回 70（CUDA），但 ROCm 上可能无意义。
- 测试覆盖：精度测试标记为 nightly，不运行在 PR CI 中，合并前无自动化验证。性能测试非回归性质。
- 与现有方案的交互：_is_mx_w4a8 检测可能误匹配其他 Quark 方案（如 W4A4），但当前分支顺序避免了冲突。
- 影响：影响范围：
- 用户：可以直接加载和运行 amd/gpt-oss-120b-w-mxfp4-a-fp8 模型，该模型是 120B 参数的 MoE 模型，量化到 W4A8 MXFP4-FP8。
- 系统：增加了 Quark 量化框架的 MoE 方案选择，不影响现有 MXFP4 或 FP8 路径。
- 团队：AMD 团队获得验证新模型和量化方案的工具，测试注册在 nightly 套件中。
- 性能：根据 PR 提供的 benchmark，TP=1 时输出 token 吞吐 853 tok/s，TPOT 平均 8.15ms。
- 风险标记：非 AITER 路径未实现，环境变量依赖，仅 MI35x 验证，nightly 测试无 CI 覆盖，gate/up 布局默认可能不兼容

关联脉络

- 暂无明显关联 PR