PR #2034 完整报告

radixark/miles

[fp8] Gate the ue8m0 weight quantizer on the backend scale format

合并时间：2026-08-05 02:04

原文链接：http://prhub.com.cn/radixark/miles/pull/2034

执行摘要

- 一句话：按后端 scale 格式门控 ue8m0 权重量化
- 推荐动作：该 PR 值得快速精读，尤其是 3 个 commit 反映的方案演进：从 DEEPGEMM_SCALE_UE8M0 门控改为真实 TE 环境变量门控，体现了“用训练侧事实约束转换侧行为”的设计思路。值得关注的设计决策：把后端能力判断（DeepGEMM 是否支持 ue8m0）与训练侧 scale 格式选择解耦。建议后续补充：环境变量缺失兜底、AMD gfx942/gfx950 分支、以及针对转换输出的回归测试。

功能与动机

PR body 指出：_quantize_param currently selects the ue8m0 weight quantizer whenever weight_block_size == [128, 128]. This gives all such checkpoints power-of-2 scales, even on backends whose GEMM consumes plain fp32 scales. 并且在 SGLang 中 ue8m0 仅由 DEEPGEMM_SCALE_UE8M0（Blackwell）启用；Miles 训练使用 NVTE_FP8_BLOCK_SCALING_FP32_SCALES=1 已产出连续 fp32 weight scales，因此非 Blackwell rollout 把训练 scale 转成 power-of-2 是不必要的，会造成精度损失。

实现拆解

变更入口是 miles/backends/megatron_utils/megatron_to_hf/processors/quantizer_fp8.py 的 _quantize_param 函数，实现按以下步骤拆解：

1. 新增环境变量依赖：文件头部新增 import os，用于读取训练侧 scale 格式约定。
2. 收紧 ue8m0 兜底分支：原逻辑在 weight_block_size == [128, 128] 且 per_block_cast_to_fp8 可用时无条件走 per_block_cast_to_fp8（产出 power-of-2 scale）；现在额外要求 os.environ["NVTE_FP8_BLOCK_SCALING_FP32_SCALES"] == "0"，即只有训练侧显式关闭 fp32 分块 scale 时才保留该路径。
3. 默认回退到连续 fp32 scale：当环境变量非 0 或条件不满足时，改用 blockwise_cast_to_fp8_triton，产出与训练侧一致的连续 fp32 block scale，shape 与 dtype 保持一致。这对 Hopper（DeepGEMM fp32 scale）、ROCm AITER（gemm_a8w8_blockscale 消费 fp32 scale）等后端避免不必要的 power-of-2 化。
4. 方案演进：PR body 原计划基于 deep_gemm_wrapper.DEEPGEMM_SCALE_UE8M0 做门控，最终在 review 中由 yueming-yuan 改为读取真实 TE 环境变量，更贴合训练侧的 scale 格式事实；AMD 分支（gfx942/gfx950）的区分未在本次落地。

配套说明：本次没有测试、配置或 schema 变更；3 个 commit 中最后一个是 review 期间的方案修正。

关键文件：

- miles/backends/megatron_utils/megatron_to_hf/processors/quantizer_fp8.py（模块 权重量化；类别 source；类型 dependency-wiring；符号 _quantize_param）：唯一改动文件，控制 FP8 权重量化时 ue8m0（power-of-2）路径的选择，直接决定导出到 SGLang 的 scale 格式。

关键符号：_quantize_param

关键源码片段

miles/backends/megatron_utils/megatron_to_hf/processors/quantizer_fp8.py

唯一改动文件，控制 FP8 权重量化时 ue8m0（power-of-2）路径的选择，直接决定导出到 SGLang 的 scale 格式。

```
def _quantize_param(args, name, weight, weight_block_size):
    """按后端 scale 格式选择 FP8 权重量化路径。

    name 必须是以 `.weight` 结尾的 Megatron 参数名，
    量化后返回 `(权重, scale_name)` 两个参数对。
    """
    assert name.endswith(".weight"), f"Expected weight parameter, got {name}"
    FP8_MIN = torch.finfo(torch.float8_e4m3fn).min
    FP8_MAX = torch.finfo(torch.float8_e4m3fn).max

    if weight_block_size is not None:
        # 显式 ue8m0 路径：由 DeepGEMM 权重 requant 判断决定，
        # 仅 Blackwell + DeepGEMM 场景启用
        if _get_scale_format(args, name, weight_block_size) == "ue8m0":
            qweight, scale = quant_weight_ue8m0(weight, weight_block_size=weight_block_size)
            scale = transform_scale_ue8m0(scale, mn=qweight.shape[-2])
        # 兜底分支：仅当训练侧显式关闭 fp32 分块 scale
        #  （NVTE_FP8_BLOCK_SCALING_FP32_SCALES == "0"）时保留 power-of-2 路径。
        # 注意：直接索引环境变量，未设置会抛 KeyError ，
        # 建议改用 os.environ.get(..., "1") 。
        # TODO: [128, 128] 这个判断比较 hacky ， 需要改进
        elif (
            os.environ["NVTE_FP8_BLOCK_SCALING_FP32_SCALES"] == "0"
            and per_block_cast_to_fp8 is not None
            and list(weight_block_size) == [128, 128]
        ):
            qweight, scale = per_block_cast_to_fp8(weight)
        else:
            qweight, scale = blockwise_cast_to_fp8_triton(weight, weight_block_size)
        scale_name = name.replace(".weight", ".weight_scale_inv")
```

```
    else:
        # per-tensor 量化：按 FP8 e4m3 动态范围计算 scale
        scale = weight.abs().max().clamp(min=1e-12).to(torch.float32) / FP8_MAX
        qweight = (weight / scale).clamp(min=FP8_MIN, max=FP8_MAX).to(torch.float8_e4m3fn)
        scale = scale.view(1)
        scale_name = name.replace(".weight", ".weight_scale")
    return [(name, qweight), (scale_name, scale)]
```

评论区精华

核心讨论集中在 PR conversation 的两条评论：

- wenchenvincent 指出 ROCm 也需要按架构分支：gfx942（Hopper 对应）应使用连续 fp32 scale，gfx950（Blackwell 对应）应使用 power-of-2 fp32 scale。当前实现未覆盖该区分，AMD 侧仍需跟进。
- yueming-yuan 在 commit d0dc7be 中把门控从 PR 初版的 deep_gemm_wrapper.DEEPGEMM_SCALE_UE8M0 改为读取真实 TE 设置（NVTE_FP8_BLOCK_SCALING_FP32_SCALES），并注明 AMD 侧需要 check。

此外 gemini-code-assist[bot] 发布了一条工具停用通告，无实质信息。最终 yueming-yuan 给出 APPROVED。

- ROCm 架构差异：gfx942 与 gfx950 应走不同 scale 分支 (design): 当前实现只按 NVTE_FP8_BLOCK_SCALING_FP32_SCALES 全局门控，无法表达 gfx942/gfx950 的架构差异，AMD 侧仍需后续处理。
- 门控条件从 DEEPGEMM_SCALE_UE8M0 改为真实 TE 环境变量 (design): 已以环境变量方案合并；AMD 侧行为待确认。

风险与影响

- 风险：
  1. 环境变量缺失直接崩溃：os.environ["NVTE_FP8_BLOCK_SCALING_FP32_SCALES"] 是直接索引，未设置时 _quantize_param 会抛 KeyError。Miles 训练通常设置该变量为 1，但独立运行的转换脚本不保证设置，存在回归风险；建议改为 os.environ.get(..., "1")。
  2. 训练 / 转换环境耦合：用训练侧环境变量 gate 转换侧行为，若转换进程与训练进程环境不一致（例如手动导出时未继承），会导致量化格式意外变化或崩溃。
  3. AMD gfx950 未覆盖：wenchenvincent 明确指出 gfx950 需要 power-of-2 scale，当前实现只区分了环境变量 0/ 非 0，无法表达 ROCm 内部架构差异。
  4. 缺少测试配套：PR 未新增任何测试，quantizer_fp8.py 是 FP8 导出核心路径，行为变化缺少回归保护。
- 影响：
  - 对非 Blackwell 后端（Hopper、sm89/sm120、ROCm）：FP8 权重量化输出的 scale 从 power-of-2 变为连续 fp32，与训练侧一致，降低不必要的精度损失；前提是转换环境设置了 NVTE_FP8_BLOCK_SCALING_FP32_SCALES=1 或未设置（但未设置会崩溃）

。

- 对 Blackwell：受 _get_scale_format 的 ue8m0 分支保护，DeepGEMM 启用时行为不变；兜底分支行为取决于环境变量，需要确认与既有 SGLang ue8m0 逻辑的一致性。
- 对团队：量化导出逻辑与训练 scale 格式约定耦合，后续引入新后端（如 gfx950）时需要扩展该门控模型。
- 风险标记：环境变量缺失会 KeyError，缺少测试覆盖，AMD gfx950 分支未实现，核心量化路径变更

关联脉络

- PR #2014 fix: quantize non-interleaved DSA indexer wk: 同一文件 miles/backends/megatron_utils/megatron_to_hf/processors/quantizer_fp8.py 的 FP8 量化分支调整，属于同一条 FP8 导出链路。
- PR #2043 Compact NVFP4 BF16 MoE exclusion metadata: 同为模型量化导出工具链（FP8/NVFP4）的元数据格式调整，可对照参考。
- PR #1571 GLM-5.2 kernel fix and GB300 training config: Blackwell 相关内核与训练配置，与本 PR 针对的 Blackwell/ 非 Blackwell scale 格式差异同属后端适配脉络。