PR #35111 完整报告

sgl-project/sglang

[AMD] diffusion: normalize ModelOpt-FP8 weights to e4m3fnuz on gfx942

合并时间：2026-08-17 17:35

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35111

执行摘要

- 一句话：gfx942 FP8 权重转 e4m3fnuz 修复 diffusion warmup 崩溃
- 推荐动作：值得精读。这是一个小而精准的平台兼容性修复：用 20 行代码解决了 AMD 侧 FP8 dtype 差异导致的确定性崩溃，核心权衡（scale 加倍保持数值等价、保留原生 fp8 路径）具有通用参考价值。关注点：normalize_e4m3fn_to_e4m3fnuz 的数值语义、is_fp8_fnuz 的判定位置（已从旧路径迁移）、以及 copy_or_rebind_param 写回顺序与 requantize 之间的隐式契约。缺点是缺少单元测试，可考虑后续补充针对归一化分支的用例。

功能与动机

29590 报告 gfx942（MI325）上所有 multimodal-gen AMD diffusion 测试自 2026-06-20 起 10/10 确定性失败，错误为 torch._scaled_mm 抛出 HIPBLAS_STATUS_NOT_SUPPORTED，阻塞所有触发 AMD diffusion 测试的 PR 合入。PR body 指出根因：gfx942 原生 fp8 dtype 是 e4m3fnuz，scaled_fp8_quant() 产出的激活为 fnuz，而 ModelOpt checkpoint（FLUX.2）导出 e4m3fn 权重，混合 dtype 的 GEMM 被 hipBLASLt 拒绝，服务器在 warmup 阶段失败。修复目标是保持 fp8 路径而非回退 bf16 反量化，因此选择在权重加载阶段做归一化。

实现拆解

1. 变更入口：python/sglang/multimodal_gen/runtime/layers/quantization/modelopt_quant.py 的 ModelOptFp8LinearMethod.process_weights_after_loading，这是所有 ModelOpt FP8 线性层权重加载后的必经处理点。
2. 新增强制 import：is_fp8_fnuz 从 sglang.kernels.ops.quantization.fp8_kernel 导入（该 helper 已从旧路径迁移，这是相对被取代的 #28889 的关键修正）；normalize_e4m3fn_to_e4m3fnuz 从 sglang.srt.layers.quantization.fp8_utils 导入。

3. 新增 fnuz 条件分支：if is_fp8_fnuz(): 时调用 normalize_e4m3fn_to_e4m3fnuz(weight, weight_scale, input_scale)，并将归一化后的 scale 通过 copy_or_rebind_param 写回 layer.weight_scale / layer.input_scale，保证后续逻辑读取到同步更新后的值。
4. 复用原有流程：requantize_with_max_scale 改用归一化后的局部 weight，第二个参数读取已更新的 layer.weight_scale（这是归一化 scale 生效的隐含契约）；后续转置绑定 layer.weight.data、convert_to_channelwise、input_scale.max() 等逻辑保持不变。
5. 配套与测试：无新增单元测试。PR body 说明现有 multimodal-gen FLUX.2 ModelOpt-FP8 CI job 在 gfx942 上覆盖该路径（修改前失败、修改后通过）；另有 2 个 commit 仅修正 isort 导入顺序，无逻辑变化。

关键文件：

- python/sglang/multimodal_gen/runtime/layers/quantization/modelopt_quant.py（模块量化层；类别 source；类型 data-contract；符号 process_weights_after_loading, normalize_e4m3fn_to_e4m3fnuz, is_fp8_fnuz）：唯一变更文件。在 ModelOptFp8LinearMethod.process_weights_after_loading 中新增 is_fp8_fnuz() 条件分支，调用 normalize_e4m3fn_to_e4m3fnuz 将 e4m3fn 权重归一化为 e4m3fnuz 并加倍 scale，修复 gfx942 warmup 时 HIPBLAS_STATUS_NOT_SUPPORTED 崩溃。

关键符号：process_weights_after_loading, normalize_e4m3fn_to_e4m3fnuz, is_fp8_fnuz

关键源码片段

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

唯一变更文件。在 ModelOptFp8LinearMethod.process_weights_after_loading 中新增 is_fp8_fnuz() 条件分支，调用 normalize_e4m3fn_to_e4m3fnuz 将 e4m3fn 权重归一化为 e4m3fnuz 并加倍 scale，修复 gfx942 warmup 时 HIPBLAS_STATUS_NOT_SUPPORTED 崩溃。

```
# ModelOpt FP8 线性层的权重后处理。
# 背景：gfx942（MI300X / MI325X）原生 fp8 为 e4m3fnuz，scaled_fp8_quant()
# 产出的激活是 fnuz；而 ModelOpt checkpoint（如 FLUX.2）导出 e4m3fn 权重，
# 两者混用会让 hipBLASLt 直接拒绝 torch._scaled_mm（HIPBLAS_STATUS_NOT_
SUPPORTED），
# 服务在 warmup 阶段崩溃，只能通过权重归一化来对齐 dtype。
from sglang.kernels.ops.quantization.fp8_kernel import is_fp8_fnuz
from sglang.srt.layers.quantization.fp8_utils import normalize_e4m3fn_to_e4m3fnuz


def process_weights_after_loading(self, layer: torch.nn.Module) -> None:
    # 仅 fnuz 硬件走归一化；非 ROCm 平台 is_fp8_fnuz() 为 False，原路径不变。
    weight = layer.weight
    if is_fp8_fnuz():
        # e4m3fn 与 e4m3fnuz 位模式只差符号位解释，重新解释并加倍 scale
        # 即可数值等价，从而把 GEMM 留在原生 fp8 路径而非回退 bf16 反量化。
        weight, weight_scale, input_scale = normalize_e4m3fn_to_e4m3fnuz(
```

```
                weight=layer.weight,
                weight_scale=layer.weight_scale,
                input_scale=layer.input_scale,
            )
            # 归一化后的 scale 写回 layer 参数是关键。
            # 后续 requantize_with_max_scale 读取 layer.weight_scale 时拿到的
            # 就是已加倍的值，保证权重与 scale 的配对关系不被破坏。
            copy_or_rebind_param(layer, "weight_scale", weight_scale)
            if input_scale is not None:
                copy_or_rebind_param(layer, "input_scale", input_scale)

        # 原有重量化流程不变：求 max scale、重量化、转置绑定回 layer.weight。
        max_w_scale, quantized_weight = requantize_with_max_scale(
            weight, layer.weight_scale, layer.logical_widths
        )
        layer.weight.data = quantized_weight.t().detach()
        layer.weight.requires_grad_(False)
        if self.cutlass_fp8_supported:
            max_w_scale = convert_to_channelwise(max_w_scale, layer.logical_widths)
        copy_or_rebind_param(layer, "weight_scale", max_w_scale)
        copy_or_rebind_param(layer, "input_scale", layer.input_scale.max())
```

评论区精华

Review 无评论，HaiShaw 直接 APPROVE。唯一讨论素材是 HaiShaw 在 #29590 的评论 'gfx942e4m3_fnuzspecificnormalization'，明确该修复是gfx942e4m3fnuz平台特有问题。PR body 中作者论证了设计权衡：*e*4m3fn 重解释为 *e*4m3fnuz 并加倍 scale 数值完全等价，可以避免回退 bf16 反量化；且 is_fp8_fnuz() 在非 ROCm 平台恒为 False，新分支不影响现有路径。

- gfx942 *e*4m3fnuz 特定归一化的设计与定位 (design): 维护者 APPROVE，确认采用平台特定归一化方案，而非通用 fallback。

风险与影响

- 风险：
  - 平台特定风险：变更仅当 is_fp8_fnuz() 为真时生效（ROCm gfx942 MI300X/MI325X），非 ROCm 平台不受影响，但需保证 is_fp8_fnuz() 判定本身可靠。
  - 数值正确性依赖：修复有效性建立在 *e*4m3fn 与 *e*4m3fnuz 位模式等价重解释的假设上；normalize_*e*4m3fn_to_*e*4m3fnuz 的实现若调整，需回归验证。
  - 隐式参数契约：requantize_with_max_scale(weight, layer.weight_scale, ...) 依赖 copy_or_rebind_param 已把归一化 scale 写回 layer.weight_scale，代码可读性上有隐性耦合。
  - 测试覆盖缺口：无新增单测，回归保护依赖现有 multimodal-gen FLUX.2 ModelOpt-FP8 CI job，若 CI 硬件矩阵变化可能失去覆盖。
- 影响：

- 用户影响：AMD gfx942 用户首次可在原生 fp8 路径上运行 FLUX.2 ModelOpt-FP8，warmup 不再崩溃；此前只能等待 hipBLASLt 支持或回退。
- 系统影响：恢复 multimodal-gen AMD CI，解除对相关 PR 的合入阻塞（#29590 持续 10+ 天）。
- 团队影响：维护者通过 APPROVE 快速合入；后续同类 FP8 dtype 差异问题可复用该归一化模式。
- 影响范围：1 个文件、20 行，逻辑仅限 ModelOpt FP8 权重加载路径。
- 风险标记：平台特定修复，无新增单元测试，数值等价性假设，核心权重加载路径

关联脉络

- PR #28889 [AMD] diffusion: normalize ModelOpt-FP8 weights to e4m3fnuz on gfx942: PR body 明确 Supersedes #28889；该 PR 因 merge 历史过长被弃，本 PR 以单 commit rebase 方式重提同一修复，并修正 is_fp8_fnuz 的 import 位置。
- PR #35020 [Fix] Correct dense FP8 Marlin bias ordering: 同属 FP8 量化正确性修复（dense FP8 Marlin bias 通道顺序），显示近期 FP8 路径在 AMD/ 通用平台的兼容性收紧趋势。
- PR #34988 [Diffusion] Reuse SRT SigLIP vision model: 同属 multimodal_gen 运行时层演进，diffusion 路径近期持续重构，本 PR 是该方向上的平台正确性补充。