

PR #56176 完整报告

vllm-project/vllm

[ROCm] [Bugfix] Enable Load and Inference of GLM-5.3-Flash Quark MXFP4 Checkpoint

合并时间: 2026-09-16 09:25

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/56176>

执行摘要

- 一句话: 修复 GLM-5.3-Flash Quark MXFP4 权重加载崩溃与 scale 名误匹配
- 推荐动作: 值得精读, 尤其对量化加载链路 (Quark、AutoWeightsLoader) 和模型目录组织感兴趣的开发者。本 PR 展示了“以 checkpoint 事实驱动修复”的典型路径: 先用 config.json 验证 layer_quant_config 的键形式, 再决定修模型映射还是修量化库, 避免在通用层引入不当行为。测试设计 (用 recording param 隔离加载路径) 也值得借鉴。

功能与动机

Quark checkpoint `amd/GLM-5.3-Flash-Quark-MXFP4` 此前完全无法加载 (先抛 `AssertionError` 再抛 `KeyError`)。PR body 明确指出“Two independent bugs break weight load”, 而 Issue #54641 复现了第一个 bug: identity entry 会让 `gate_up_proj` 静默回退到 `global_quant_config` (MXFP4), 与 checkpoint 中 block-FP8 张量形状不匹配。第二个 bug 是 MLA 投影 scale 名不一致: Quark 导出 `weight_scale`, 而加载器只认 `weight_scale_inv`。

实现拆解

1. 变更入口: 核心修复集中在 `vllm/models/glm5next/common/model.py` (由 `nvidia/model.py` 移动而来)。- 在 `Glm5NextForConditionalGeneration` 上覆盖 `packed_modules_mapping = {"gate_up_proj": ["gate_proj", "up_proj"]}`, 替换从 `Glm4vForConditionalGeneration` 继承的 identity 映射。GLM-5.3-Flash 的 checkpoint (Quark、NVFP4、zai 基座) 均以分离张量存储 gate/up, 展开为真实 shard 名后, Quark 的 `layer_quant_config` 才能命中 per-layer block-FP8 方案, 避免回退到全局 MXFP4。- `_try_load_fp8_attn_proj` 的 scale 识别从 `is_scale = "weight_scale_inv" in name` 扩展为 `"weight_scale_inv" in name or name.endswith(".weight_scale")`, 使 Quark 命名的 scale 进入同一套 buffer 配对 + block 反量化逻辑。
2. 目录重构: `model.py`、`attention.py`、`kda.py`、`mtp.py`、`multimodal.py` 从 `glm5next/nvidia/` 平移至 `glm5next/common/`, `glm5next/__init__.py` 的导入同步改为 `common` 路径; `mtp.py` 中对 `fused_eh_norm` 的导入改为显式 `vllm.models.glm5next.nvidia.ops.fused_eh_norm` (ops 目录仍留在 `nvidia` 下)。该调整由 review 推动, 目的是表达修复对 AMD/NVIDIA 都适用。
3. 测试配套: `tests/quantization/test_quark.py` 新增 4 个测试和辅助设施 (`_glm5_mixed_precision_config`、`_GLM5RecordingParam`、`_glm5_block_fp8`), 覆盖映

射真实性、scheme 决议、Quark weight_scale 识别以及 DeepSeek weight_scale_inv 回归；测试不依赖完整模型实例，通过 recording param 验证加载路径正确性。

4. 验证：作者在 4x AMD MI355X 上以 VLLM_ROCM_USE_AITER=1 服务 Quark checkpoint, gsm8k 5-shot strict exact_match 0.9219；同时在 H100 上验证 NVFP4 变体不受影响并受益（0.9371）。

关键文件：

- vllm/models/glm5next/common/model.py（模块 模型加载；类别 source；类型 core-logic；符号 Glm5NextForConditionalGeneration.packed_modules_mapping, _try_load_fp8_attn_proj）：承载两处核心修复：覆盖 packed_modules_mapping 使 gate/up 展开为真实 shard 名，以及 _try_load_fp8_attn_proj 接受 Quark 的 weight_scale scale 名。文件同时由 nvidia/model.py 移动至 common/。
- tests/quantization/test_quark.py（模块 量化；类别 test；类型 test-coverage；符号 _glm5_mixed_precision_config, _GLM5RecordingParam, _glm5_block_fp8, test_glm5next_gate_up_proj_mapping_is_genuine_fusion）：新增 145 行回归测试，用 recording param 隔离验证映射解析、scheme 决议、Quark scale 识别与 DeepSeek scale 回归，是本次修复正确性的关键保障。
- vllm/models/glm5next/__init__.py（模块 模型入口；类别 source；类型 data-contract）：模型入口导入从 nvidia/ 切换到 common/，是目录重构的关键接线点，保证 Glm5NextForCausalLM 等符号仍可从包顶层导入。
- vllm/models/glm5next/common/mtp.py（模块 推测解码；类别 source；类型 rename-or-move）：随目录重构移动；import 调整（fused_eh_norm 显式指向仍留在 nvidia 下的 ops 路径）是编译可用的关键配套。
- vllm/models/glm5next/common/attention.py（模块 注意力；类别 source；类型 rename-or-move）：纯移动：GLM-5.3-Flash 的 MLA/ 稀疏注意力实现随共享化迁移到 common，无内容变更。
- vllm/models/glm5next/common/kda.py（模块 模型代码；类别 source；类型 rename-or-move）：纯移动：KDA 相关实现随共享化迁移到 common，无内容变更。
- vllm/models/glm5next/common/multimodal.py（模块 多模态；类别 source；类型 rename-or-move）：纯移动：多模态处理部分随共享化迁移到 common，无内容变更。
- vllm/models/glm5next/common/__init__.py（模块 模型入口；类别 source；类型 data-contract）：新增 common 包的初始化文件，支撑目录重构后的包结构。

关键符号：Glm5NextForConditionalGeneration.packed_modules_mapping, _try_load_fp8_attn_proj, _glm5_mixed_precision_config, test_glm5next_gate_up_proj_mapping_is_genuine_fusion, test_glm5next_genuine_fusion_resolves_gate_up_to_block_fp8, test_glm5next_attn_loader_accepts_quark_weight_scale, test_glm5next_attn_loader_accepts_deepseek_weight_scale_inv

关键源码片段

vllm/models/glm5next/common/model.py

承载两处核心修复：覆盖 `packed_modules_mapping` 使 `gate/up` 展开为真实 shard 名，以及 `_try_load_fp8_attn_proj` 接受 Quark 的 `weight_scale` scale 名。文件同时由 `nvidia/model.py` 移动至 `common/`。

```
class Glm5NextForConditionalGeneration(Glm4vForConditionalGeneration, HasInnerState,
IsHybrid, MixtureOfExperts):
```

```
    # GLM-5.3-Flash 把 dense-MLP 的 gate/up 存成两个独立张量,
    # 与 Glm4vMoeForConditionalGeneration、glm4_moe、deepseek_v2 一致。
    # 因此 fused `gate_up_proj` 必须展开成真实 shard 名 `gate_proj` / `up_proj`,
    # Quark 的 per-layer 量化配置 (layer_quant_config) 才能命中。
    # 从 Glm4vForConditionalGeneration 继承的 identity 映射只适用
    # GLM-4.1V 那种预融合 checkpoint, 会让层回退到全局 MXFP4 方案,
    # 加载 FP8 权重时因形状不匹配而崩溃 (即 #54641 复现的 AssertionError) 。
    packed_modules_mapping = {
        "gate_up_proj": ["gate_proj", "up_proj"],
    }
```

```
def _try_load_fp8_attn_proj(
```

```
    name,
    tensor,
    buf,
    params_dict,
    loaded_params,
    kv_a_pad_size: int,
```

```
) -> bool:
```

```
    # 仅处理 checkpoint 里以 block-FP8 存储、但模型侧保持 BF16 的
    # q_a_proj / kv_a_proj_with_mqa / q_b_proj / o_proj 投影;
    # 返回 True 表示已接管该 tensor 的加载。
```

```
    matched = None
```

```
    for suffix, info in _FP8_ATTN_PROJS.items():
```

```
        if suffix in name:
            matched = (suffix, info)
            break
```

```
    if matched is None:
```

```
        return False
```

```
    suffix, (key, target_base, shard_id, is_kva) = matched
```

```
    is_weight = name.endswith(".weight") and tensor.dtype == torch.float8_e4m3fn
```

```
    # DeepSeek 原生 scale 名是 weight_scale_inv, Quark checkpoint 导出的是
```

```
    # weight_scale; 这里两者都接受, 统一进入下面的 buffer 配对逻辑。
```

```
    is_scale = "weight_scale_inv" in name or name.endswith(".weight_scale")
```

```
    if not is_weight and not is_scale:
```

```
        return False
```

```
    layer_prefix = name.rsplit(suffix, 1)[0]
```

```
    target_w = f"{layer_prefix}.{target_base}.weight"
```

```
    target_s = f"{layer_prefix}.{target_base}.weight_scale_inv"
```

```
    # 若模型侧真的保留了 FP8 (存在 weight_scale_inv 参数),
```

```
    # 说明不该反量化, 直接返回 False 交给常规 stacked/direct 加载路径。
```

```

if target_s in params_dict:
    return False

# weight 与 scale 到达顺序不固定，先到的一半进 buf 缓冲，
# 等另一半到齐再一次性反量化，避免破坏 fused_qkv_a_proj 的 shard 语义。
entry = buf.setdefault(layer_prefix, {}).setdefault(key, {})
entry["weight" if is_weight else "scale"] = tensor
if "weight" not in entry or "scale" not in entry:
    return True

weight_fp8, scale_inv = entry["weight"], entry["scale"]
buf[layer_prefix].pop(key, None)
block_size = weight_fp8.shape[1] // scale_inv.shape[1]
weight_bf16 = _dequant_fp8_block(weight_fp8, scale_inv, block_size)
# NoPE 场景下 kv_a 需要补零 rope 段（宽度对齐），再写入目标参数。
if is_kva and kv_a_pad_size > 0:
    pad = torch.zeros(
        kv_a_pad_size,
        weight_bf16.shape[1],
        dtype=weight_bf16.dtype,
        device=weight_bf16.device,
    )
    weight_bf16 = torch.cat([weight_bf16, pad], dim=0)

param = params_dict[target_w]
# 随后调用 param.weight_loader(param, weight_bf16, shard_id) 写入目标：
# shard_id 决定 fused_qkv_a_proj 内部是 q_a (0) 还是 kv_a (1) 分支；
# 完成后把 target_w 加进 loaded_params，并返回 True。

```

tests/quantization/test_quark.py

新增 145 行回归测试，用 recording param 隔离验证映射解析、scheme 决议、Quark scale 识别与 DeepSeek scale 回归，是本次修复正确性的关键保障。

```

# 全局 MXFP4、dense-MLP 的 gate/up 单独标记 block-FP8 的混合精度配置，
# 复刻 amd/GLM-5.3-Flash-Quark-MXFP4 的 layer_quant_config 形状。
def _glm5_mixed_precision_config() -> QuarkConfig:
    fp8 = {"weight": _GLM5_BLOCK_FP8_WEIGHT, "input_tensors": _GLM5_BLOCK_FP8_INPUT}
    return QuarkConfig(
        {
            "global_quant_config": {"weight": _GLM5_MXFP4_WEIGHT, "input_tensors": None},
            "layer_type_quant_config": {},
            "layer_quant_config": {"*mlp.gate_proj": fp8, "*mlp.up_proj": fp8},
            "exclude": [],
        }
    )

def test_glm5next_genuine_fusion_resolves_gate_up_to_block_fp8():
    # fused 模块必须展开成真实 shard 名，每个 shard 才能命中

```

```

# per-layer 的 block-FP8 条目，而不是回退到全局 MXFP4 方案。
config = _glm5_mixed_precision_config()
config.packed_modules_mapping = {"gate_up_proj": ["gate_proj", "up_proj"]}
_, _, scheme_cls = config.get_scheme_cls(LinearBase, _GLM5_GATE_UP)
assert scheme_cls is QuarkW8A8Fp8PerBlock

def test_glm5next_attn_loader_accepts_quark_weight_scale():
    # 回归 KeyError (...kv_a_proj_with_mqa.weight_scale) : fused q_a/kv_a
    # 保持 BF16 并在加载时反量化，Quark 的 scale 名必须被识别，
    # 且路由到 fused_qkv_a_proj 的正确 shard。
    from vllm.models.glm5next.common.model import _dequant_fp8_block, _try_load_fp8_attn_proj

    prefix = "layers.3.self_attn"
    target = _GLM5RecordingParam()
    params_dict = {f"{prefix}.fused_qkv_a_proj.weight": target}
    buf, loaded = {}, set()
    weight, scale = _glm5_block_fp8(256, 256)

    # fp8 weight 先到达 -> 只进缓冲，此时还未触发加载。
    assert _try_load_fp8_attn_proj(
        f"{prefix}.kv_a_proj_with_mqa.weight", weight, buf, params_dict, loaded, 0
    )
    assert target.loaded_weight is None
    # Quark 命名的 scale 补齐配对 -> 反量化并加载。
    assert _try_load_fp8_attn_proj(
        f"{prefix}.kv_a_proj_with_mqa.weight_scale", scale, buf, params_dict, loaded, 0
    )
    assert target.loaded_weight is not None
    assert target.loaded_weight.dtype == torch.bfloat16
    assert target.loaded_shard == 1 # kv_a 是 fused_qkv_a_proj 的 shard 1
    assert torch.equal(target.loaded_weight, _dequant_fp8_block(weight, scale, 128))
    assert f"{prefix}.fused_qkv_a_proj.weight" in loaded

```

评论区精华

1. #54641 的修复范围争议：fxmarty-amd 质疑“identity 映射导致 layer_quant_config 失效”是合法且更普遍的问题，询问本 PR 为何不直接在 quark.py 修复，并要求补测试。
ColinZ22 回应：本 PR 只解决该 issue 中复现的 GLM-5.3-Flash 实际阻塞（reproduction 中的崩溃），checkpoint 的 layer_quant_config 用的是 split 名称，因此模型内 override 是正确路径；liulanze 对照发布版 config.json 确认没有 gate_up_proj 键，佐证了该决策。
2. 目录结构讨论：BowenBao 转述 AndreasKaratzas 建议，要求把 model.py 移到 common/，因为修复在 amd/nvidia 通用；作者照做并 rebase 后重新验证两个 checkpoint 的 GSM8k。

3. NV 适用范围确认: AndreasKaratzas 分别对 packed_modules_mapping 和 is_scale 两处改动问“是否同样适用于 NV”。作者确认 nvidia/GLM-5.3-Flash-NVFP4 也存分离 gate/up 张量 (H100 gsm8k 0.9371), 但 NVFP4 的 MLA 投影已在 checkpoint 内反量化为 BF16, 不会进入 _try_load_fp8_attn_proj 路径。
- 54641 的修复范围与测试要求 (question): ColinZ22 澄清本 PR 只解决 issue 中复现的 GLM-5.3-Flash 实际阻塞, checkpoint 的 layer_quant_config 使用 split 名称, 模型内 override 才是正确修复; liulanze 对照发布版 config.json 确认无 gate_up_proj 键。测试随后补充进 test_quark.py。
 - 共享文件移至 common 目录 (design): 作者将 model.py、attention.py、kda.py、mtp.py、multimodal.py 全部移入 common/, rebased 后重新验证两个 checkpoint 的 GSM8k。
 - NV 平台适用范围确认 (question): 作者确认 nvidia/GLM-5.3-Flash-NVFP4 同样分离存储 gate/up (H100 gsm8k 0.9371); 而 NVFP4 的 MLA 投影已在 checkpoint 内反量化为 BF16, 不会进入 _try_load_fp8_attn_proj 路径。

风险与影响

- 风险:
 1. 量化方案解析敏感性: packed_modules_mapping 覆盖只对 Glm5NextForConditionalGeneration 生效, 不影响继承源 Glm4vForConditionalGeneration, 但若未来出现 GLM-5.3-Flash 变体以预融合 gate_up_proj 存储, 本修复会导致 shard 展开后找不到目标名, 需要 checkpoint 事实驱动演进。
 2. scale 匹配放宽: name.endswith(".weight_scale") 比之前的子串匹配更宽, 虽被限制在 _FP8_ATTN_PROJS 后缀内, 但若 checkpoint 同时携带其他语义的 weight_scale 张量, 可能被误当作反量化 scale 缓冲。
 3. 目录重构影响: nvidia/ 与 common/ 的移动会破坏仓库外部对旧路径的直接 import; 仓库内部导入均已更新, 但需要 CI (Buildkite #89206) 覆盖确认。
 4. 与 #54702 的方案分歧: 若日后在 quark.py 中加入 identity 映射容忍逻辑, 与本 PR 的模型内 override 可能产生双重解析, 需要协调语义。- 影响: 影响范围集中在 GLM-5.3-Flash 系列模型: AMD Quark MXFP4 checkpoint 从完全不可用变为可加载、可服务, gsm8k 精度与源 checkpoint 对齐 (0.9219 vs 0.9121); NVIDIA NVFP4 变体不受回归影响并在 H100 上验证达标。对团队而言, common/ 目录的建立明确了 glm5next 代码跨厂商共享的组织方式, 降低后续 ROCm/NVIDIA 双线维护成本。对 vLLM 其他模型无直接行为影响。- 风险标记: 量化方案解析敏感性, scale 名称匹配放宽, 跨厂商目录重构, 仅覆盖 Glm5Next 路径

关联脉络

- PR #54702 Quark: tolerate identity entries in packed_modules_mapping: 该 PR 是对 Issue #54641 的另一种修复尝试 (在 quark.py 中容忍 identity 映射), 但按 checkpoint 实际 key (split names) 仍会回退到 MXFP4; 本 PR 明确不采用该方向, 而是模型内 override。

- PR #54556 GLM-5.3: remap weight_scale name in deepseek_v2.py: 与 Bug 2 相似的 scale 命名问题, 但针对完整 GLM-5.3 (GlmMoeDsaForCausalLM) 路径, 不覆盖 GLM-5.3-Flash; 本 PR 在 glm5next 自己的加载器中修复。
- PR #54770 QuarkW8A8Fp8PerBlock registers weight_scale: 使量化模块可原生加载 weight_scale, 但 GLM-5.3-Flash 的 MLA 投影保持 BF16、由 _try_load_fp8_attn_proj 手动反量化, 因此本 PR 仍需单独识别 Quark scale 名。