PR #36061 完整报告

sgl-project/sglang

[Diffusion] Dispatch mixed Comfy NVFP4 and INT8 layers

合并时间：2026-08-25 09:19

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/36061

执行摘要

- 一句话：实现 Comfy 混合 NVFP4/INT8 检查点按层分发量化
- 推荐动作：值得精读。set_comfy_layer_markers 展示了如何在一份量化配置内按层组合两种量化方法，get_quant_method 的 prefix 分发模式也可复用于其他混合量化检查点。注意本 PR 栈式叠加在 #36044 之上，建议结合父 PR 与 #36066 一起阅读，理解 Diffusion 量化加载的完整演进方向。

功能与动机

PR body 明确指出：社区导出越来越多地把 NVFP4 MLPs 与 INT8 ConvRot attention 混在同一个 safetensors 文件中；父 PR（#36044）已经能识别 NVFP4 布局，但遇到混合标记集时会直接拒绝，而不是尊重每个序列化层自己的格式。本 PR 的目标是让加载器按层标记分发量化实现，复用现有 ModelOpt NVFP4 与 Kitchen INT8 代码，用户无需显式选择量化方式。

实现拆解

1. 扩展 NVFP4 配置以承载 INT8 子配置（modelopt_quant.py）：ModelOptFp4Config 新增 _comfy_int8_config 属性与 set_comfy_layer_markers 方法，校验 layer_markers 中只允许 nvfp4 与 int8_tensorwise 两种格式，并收集 int8 标记构造 KitchenInt8Config。get_quant_method 对命中 INT8 标记的 prefix 委托给 KitchenInt8Config，其余层走原 NVFP4 路径，从而在一份配置内实现跨量化方法的分发。
2. 放宽检查点量化解析分支（minimax_h3_weights.py）：resolve_minimax_h3_checkpoint_quantization 将进入 NVFP4 路径的条件从 formats == 与“nvfp4” 改为 “nvfp4” in formats，使混合标记集也能进入；对白名单外的伴随格式提前抛出 NotImplementedError；用 isinstance(config, ModelOptFp4Config) 替代原 None 检查，并在返回前调用 set_comfy_layer_markers 注入逐层标记。
3. 新增逐层分发测试（test_transformer_quant.py）：test_minimax_h3_mixed_nvfp4_int8_dispatches_each_layer 构造同时含 nvfp4（qkv_proj）与 int8_tensorwise（out_proj，convrot_groupsize=256）标记的 safetensors，在 mock 设备能力后断言 get_quant_method 分别返回 ModelOptFp4LinearMethod 与 KitchenInt8LinearMethod。
4. 文档补充（MiniMax-H3.mdx）：在 cookbook 的量化小节说明混合文件中标记为 int8_tensorwise 的层会自动分发到 Kitchen INT8 ConvRot 路径。

测试配套方面，单元测试覆盖了标记解析、格式白名单校验与逐层分发；测试中 mock 了 current_platform.get_device_capability 与 kitchen_int8._load_comfy_kitchen，未覆盖真实 kernel 加载。本次无配置项、部署或 schema 变更。

关键文件：

- python/sglang/multimodal_gen/runtime/layers/quantization/modelopt_quant.py（模块 量化配置；类别 source；类型 data-contract；符号 ModelOptFp4Config, set_comfy_layer_markers, get_quant_method）：核心变更文件。在 ModelOptFp4Config 中新增 set_comfy_layer_markers 与 _comfy_int8_config，并在 get_quant_method 中实现按 prefix 向 Kitchen INT8 量化方法分发，是混合检查点支持的数据契约核心。
- python/sglang/multimodal_gen/runtime/loader/minimax_h3_weights.py（模块 权重加载；类别 source；类型 dependency-wiring；符号 resolve_minimax_h3_checkpoint_quantization）：修改 MiniMax-H3 检查点量化解析分支，从只接受纯 NVFP4 标记集扩展为接受含 nvfp4 的混合标记集，并在构建配置后注入逐层标记，是整个功能的接线点。
- python/sglang/multimodal_gen/test/unit/test_transformer_quant.py（模块 量化测试；类别 test；类型 test-coverage；符号 test_minimax_h3_mixed_nvfp4_int8_dispatches_each_layer）：新增混合 NVFP4/INT8 逐层分发测试，覆盖格式解析与 get_quant_method 分发两个关键环节，保护新契约不回归。
- docs/cookbook/diffusion/MiniMax/MiniMax-H3.mdx（模块 文档；类别 docs；类型 documentation）：补充混合检查点行为说明，告知用户 int8_tensorwise 层会自动分发到 Kitchen INT8 ConvRot 路径，属于功能配套文档。

关键符号：set_comfy_layer_markers, ModelOptFp4Config.get_quant_method, resolve_minimax_h3_checkpoint_quantization, test_minimax_h3_mixed_nvfp4_int8_dispatches_each_layer

关键源码片段

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

核心变更文件。在 ModelOptFp4Config 中新增 set_comfy_layer_markers 与 _comfy_int8_config，并在 get_quant_method 中实现按 prefix 向 Kitchen INT8 量化方法分发，是混合检查点支持的数据契约核心。

```
# modelopt_quant.py 中的关键片段（节选）
class ModelOptFp4Config(ModelOptQuantConfig):
    '''NVFP4 量化配置类。

    在 Comfy 混合检查点场景下，一份 safetensors 里可能同时包含
    NVFP4 MLP 与 INT8 ConvRot attention 两种格式，因此该类额外
    持有一个 Kitchen INT8 子配置，并按 layer prefix 将层分发给
    对应的量化实现。
    '''

    def __init__(self, ...):
```

```
        ...
        # 新增：保存从 layer_markers 提取的 INT8 子配置；
        # 为 None 时表示纯 NVFP4 检查点，行为与旧版本一致
        self._comfy_int8_config: KitchenInt8Config | None = None

    def set_comfy_layer_markers(self, layer_markers: dict[str, dict[str, Any]]) -> None:
        '''接收 checkpoint 的按层格式标记，并校验支持的格式集合。'''
        # 白名单只有 nvfp4 与 int8_tensorwise，出现其他格式直接报错，
        # 避免带着未处理的格式进入后续权重加载
        unsupported = {
            str(marker.get('format')) for marker in layer_markers.values()
        } - {'nvfp4', 'int8_tensorwise'}
        if unsupported:
            raise ValueError(
                'NVFP4 checkpoints cannot dispatch companion Comfy formats: '
                + ', '.join(sorted(unsupported))
            )
        # 收集所有 int8_tensorwise 层并构造 Kitchen INT8 子配置，
        # 供 get_quant_method 按 prefix 精确分发
        int8_markers = {
            prefix: marker
            for prefix, marker in layer_markers.items()
            if marker.get('format') == 'int8_tensorwise'
        }
        self._comfy_int8_config = (
            KitchenInt8Config(layer_markers=int8_markers) if int8_markers else None
        )

    def get_quant_method(self, layer: torch.nn.Module, prefix: str):
        # INT8 标记层优先委托给 Kitchen INT8 实现，其余层仍走 NVFP4 路径
        if (
            self._comfy_int8_config is not None
            and prefix in self._comfy_int8_config.layer_markers
        ):
            return self._comfy_int8_config.get_quant_method(layer, prefix)
        return self._get_quant_method(layer, prefix, Linear=ModelOptFp4LinearMethod)
```

python/sglang/multimodal_gen/runtime/loader/minimax_h3_weights.py

修改 MiniMax-H3 检查点量化解析分支，从只接受纯 NVFP4 标记集扩展为接受含 nvfp4 的混合标记集，并在构建配置后注入逐层标记，是整个功能的接线点。

```
# minimax_h3_weights.py：解析 MiniMax-H3 检查点的量化配置（节选）
def resolve_minimax_h3_checkpoint_quantization(
    layer_markers: dict[str, dict[str, Any]],
    safetensors_list: list[str] | None = None,
    param_names_mapping: dict | None = None,
    reverse_param_names_mapping: dict | None = None,
) -> QuantizationConfig | None:
    # 只要包含 nvfp4 标记就进入 NVFP4 构建路径，
```

```
    # 这样混合标记集（nvfp4 + int8_tensorwise）也能被解析
    formats = {str(marker.get('format')) for marker in layer_markers.values()}
    if 'nvfp4' in formats:
        # 白名单外的伴随格式直接拒绝，宁可报错也不要静默误解析
        unsupported = formats - {'nvfp4', 'int8_tensorwise'}
        if unsupported:
            raise NotImplementedError(
                'Unsupported Comfy NVFP4 companion format(s): '
                + ', '.join(sorted(unsupported))
            )
        if safetensors_list is None:
            raise ValueError('MiniMax-H3 NVFP4 metadata requires checkpoint files')
        config = build_nvfp4_config_from_safetensors_list(
            safetensors_list,
            param_names_mapping,
            reverse_param_names_mapping,
        )
        # 用 isinstance 校验返回类型，防止将来返回其他配置类型时静默出错
        if not isinstance(config, ModelOptFp4Config):
            raise ValueError('Could not resolve MiniMax-H3 NVFP4 checkpoint layout')
        # 把逐层标记注入配置，使 NVFP4 config 内部能够按层分发 INT8 实现
        config.set_comfy_layer_markers(layer_markers)
        config.checkpoint_uses_comfy_quantization = True
        config.checkpoint_uses_native_qkv_layout = True
        config.checkpoint_weight_scale_layout = 'swizzled'
        config.swap_weight_nibbles = True
        return config
    # 非 NVFP4 场景继续走通用 Comfy 量化解析
    return resolve_comfy_checkpoint_quantization(layer_markers)
```

评论区精华

该 PR 没有 reviewer 评论，唯一评论来自 mintlify bot 的文档预览部署通知，指向 MiniMax-H3 cookbook 的 Mintlify Preview，无实质技术讨论。技术决策主要体现在 commit 历史中：例如“Fix mixed NVFP4 dispatch test capability”与“Mock optional INT8 kernel in mixed quant test”说明作者处理了设备能力与可选 kernel 的 mock 问题；多次“merge encoder quant marker fix”提交显示编码器量化标记的合并逻辑经过了多轮修复。

- Mintlify 文档预览部署 (other): 无需要处理的技术结论；PR 无 reviewer 评论，最终由作者直接合并。

风险与影响

- 风险：核心风险集中在分发与平台依赖两方面：get_quant_method 依赖 prefix 在 _comfy_int8_config.layer_markers 中的精确匹配，若 checkpoint 标记前缀与模型内部 prefix 不一致，INT8 层会被静默降级到 NVFP4 路径，权重解析可能错位；Kitchen INT8 ConvRot kernel 并非所有平台可用，测试中 mock 了 _load_comfy_kitchen，真实加载失败场景未被 CI 覆盖；NVFP4 路径本身要求 SM100+。另外 resolve 逻辑收紧后，未来出

现白名单外的伴随格式会直接 NotImplementedError，需要同步扩展白名单。影响面局限在 Diffusion 运行时 MiniMax-H3 加载路径，不涉及 SRT 核心调度与推理路径。

- 影响：用户可直接加载社区导出的混合量化 MiniMax-H3 safetensors 而无需指定 --quantization；纯 NVFP4 检查点无 INT8 标记时 _comfy_int8_config 为 None，行为与旧版本一致。系统侧新增一层按层分发逻辑，但只影响含 int8_tensorwise 标记的检查点。团队侧为后续支持更多 Comfy 伴随格式（如 FP8，见 #36066）提供了可复用的分发框架。
- 风险标记：量化配置契约变更，平台与 kernel 依赖未全覆盖，prefix 精确匹配分发，混合格式白名单较窄

关联脉络

- PR #36066 [diffusion] feat: dispatch fp8 companions in mixed nvfp4 checkpoints: 同一主文件（modelopt_quant.py、test_transformer_quant.py）上的后续演进，将混合 NVFP4 检查点的伴随格式分发从 INT8 扩展到 FP8。
- PR #36035 [Diffusion] Add component-scoped quantization overrides: 同属 diffusion 量化加载链路，为不同组件提供量化覆盖配置，与本 PR 的 per-layer 分发能力互补。
- PR #36055 [Diffusion] Load MiniMax H3 GGUF text encoders: 同属 MiniMax-H3 检查点加载与量化配置解析链路，扩展了 text encoder 的量化格式支持。