PR #36066 完整报告

sgl-project/sglang

[diffusion] feat: dispatch fp8 companions in mixed nvfp4 checkpoints

合并时间：2026-08-25 11:26

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/36066

执行摘要

- 一句话：混合 NVFP4 检查点支持 FP8 伴随层自动调度
- 推荐动作：值得精读。该 PR 展示了如何在不破坏既有 NVFP4 路径的前提下，通过配置层扩展（数据契约 + 分发逻辑）优雅支持混合量化伴随层。关注 set_comfy_layer_markers 的格式白名单设计与 get_quant_method 的委托模式，是典型的多格式量化分派实现，可复用。同时注意其依赖链（#36060、#36061）对 FP8 输入 scale 推断的假设。

功能与动机

PR body 明确说明：'allow self-describing NVFP4 checkpoints to contain FP8 companion linears'，目标是覆盖 H3 混合导出场景，例如 NVFP4 MLP 加上动态 FP8 attention，而无需文件名规则或显式量化标志。'Dependencies Stacked on #36061 and includes #36060 because the target community checkpoints omit FP8 input scales and therefore require activation-scheme inference.' 即社区导出的检查点省略了 FP8 输入 scale，因此必须依赖先前 PR 中的激活方案推断能力。

实现拆解

1. 扩展量化配置数据契约：在 python/sglang/multimodal_gen/runtime/layers/quantization/modelopt_quant.py 中，为 ModelOptFp4Config 新增 _comfy_fp8_config 字段（类型为 ComfyFp8Config | None），并在 set_comfy_layer_markers 中把 float8_e4m3fn 加入支持格式集合，同时从 layer_markers 中筛出 FP8 标记并构造 ComfyFp8Config。
2. 路由逻辑扩展：在 ModelOptFp4Config.get_quant_method 中，在 NVFP4 默认路径之前增加 FP8 伴随层的检查：若前缀命中 self._comfy_fp8_config.layer_markers，则委托给 ComfyFp8Config.get_quant_method 返回 Fp8LinearMethod；INT8 的 Kitchen 路由保持不变。
3. 检查点格式白名单放宽：在 python/sglang/multimodal_gen/runtime/loader/minimax_h3_weights.py 的 resolve_minimax_h3_checkpoint_quantization 中，将允许的伴随格式从 {nvfp4, int8_tensorwise} 扩展为 {nvfp4, int8_tensorwise, float8_e4m3fn}，避免对 FP8 伴随层抛出 NotImplementedError。
4. 测试覆盖：在 python/sglang/multimodal_gen/test/unit/test_transformer_quant.py 中将原 test_minimax_h3_mixed_nvfp4_int8_dispatches_each_layer 重命名为 test_minimax_h3_mixed_nvfp4_companions_dispatch_each_layer，并在 metadata 与权重中增加 blocks.0.mlp.fc1 的 FP8 层，断言其被正确调度为 Fp8LinearMethod。

5. 文档更新：在 docs/cookbook/diffusion/MiniMax/MiniMax-H3.mdx 与 docs/docs/sglang-diffusion/quantization.mdx 中同步说明混合检查点可包含 INT8 或动态 / 静态 FP8 伴随层，并自动调度到对应路径。

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.init, ModelOptFp4Config.set_comfy_layer_markers, ModelOptFp4Config.get_quant_method）：核心实现：为 ModelOptFp4Config 新增 FP8 伴随层配置与分发逻辑，是本次功能的主入口。
- python/sglang/multimodal_gen/runtime/loader/minimax_h3_weights.py（模块 权重加载；类别 source；类型 core-logic；符号 resolve_minimax_h3_checkpoint_quantization）：检查点格式白名单放宽，决定混合 NVFP4 检查点能否被接受并进入后续解析流程。
- 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_companions_dispatch_each_layer）：测试覆盖：验证 NVFP4 + INT8 + FP8 混合检查点按层分发到对应量化方法，保证回归。
- docs/cookbook/diffusion/MiniMax/MiniMax-H3.mdx（模块 模型文档；类别 other；类型 core-logic）：用户文档：说明混合检查点可包含 FP8 伴随层并自动调度。
- docs/docs/sglang-diffusion/quantization.mdx（模块 量化文档；类别 other；类型 core-logic）：量化总览文档同步更新，明确 Comfy 标记支持 NVFP4 + INT8/FP8 伴随层。

关键符号：ModelOptFp4Config.set_comfy_layer_markers, ModelOptFp4Config.get_quant_method, resolve_minimax_h3_checkpoint_quantization, test_minimax_h3_mixed_nvfp4_companions_dispatch_each_layer

关键源码片段

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

核心实现：为 ModelOptFp4Config 新增 FP8 伴随层配置与分发逻辑，是本次功能的主入口。

```
# modelopt_quant.py 中 ModelOptFp4Config 的关键改动片段

class ModelOptFp4Config(ModelOptQuantConfig):
    """Config class for NVFP4."""

    def __init__(self, ...):
        super().__init__(exclude_modules, packed_modules_mapping)
        ...
        self._comfy_int8_config: KitchenInt8Config | None = None
        # 新增：用于承载混合检查点里的 FP8 伴随层配置
        self._comfy_fp8_config: ComfyFp8Config | None = None

    def set_comfy_layer_markers(self, layer_markers: dict[str, dict[str, Any]]) -> None:
        # 白名单中加入 float8_e4m3fn，允许 NVFP4 检查点携带 FP8 伴随层
```

```
        unsupported = {
            str(marker.get("format")) for marker in layer_markers.values()
        } - {"nvfp4", "int8_tensorwise", "float8_e4m3fn"}
        if unsupported:
            raise ValueError(
                "NVFP4 checkpoints cannot dispatch companion Comfy formats: "
                + ", ".join(sorted(unsupported))
            )
        # INT8 伴随层仍走 KitchenInt8Config
        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
        )
        # 新增：筛出 FP8 伴随层并构造 ComfyFp8Config
        fp8_markers = {
            prefix: marker
            for prefix, marker in layer_markers.items()
            if marker.get("format") == "float8_e4m3fn"
        }
        self._comfy_fp8_config = ComfyFp8Config(fp8_markers) if fp8_markers else None

    def get_quant_method(self, layer: torch.nn.Module, prefix: str):
        # 原有 INT8 分发保持不变
        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)
        # 新增：FP8 伴随层分发到 Comfy FP8 实现
        if (
            self._comfy_fp8_config is not None
            and prefix in self._comfy_fp8_config.layer_markers
        ):
            return self._comfy_fp8_config.get_quant_method(layer, prefix)
        # 其余层仍走 NVFP4 默认路径
        return self._get_quant_method(layer, prefix, Linear=ModelOptFp4LinearMethod)
```

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

检查点格式白名单放宽，决定混合 NVFP4 检查点能否被接受并进入后续解析流程。

```
# minimax_h3_weights.py 中 resolve_minimax_h3_checkpoint_quantization 的关键判断

def resolve_minimax_h3_checkpoint_quantization(
    layer_markers: dict[str, dict[str, Any]],
    safetensors_list: list[str] | None = None,
```

```
    ...
) -> QuantizationConfig | None:
    formats = {str(marker.get("format")) for marker in layer_markers.values()}
    if "nvfp4" in formats:
        # 允许 NVFP4 主格式携带 INT8 或 FP8 伴随层
        unsupported = formats - {"nvfp4", "int8_tensorwise", "float8_e4m3fn"}
        if unsupported:
            raise NotImplementedError(
                "Unsupported Comfy NVFP4 companion format(s): "
                + ", ".join(sorted(unsupported))
            )
        ...
        config.set_comfy_layer_markers(layer_markers)
        ...
```

评论区精华

该 PR 没有 review 评论或讨论线程。唯一的评论是 mintlify bot 的文档预览部署通知，无技术讨论。

- 无技术 review 讨论 (other): 无待解决的技术疑虑，由作者自行合并。

风险与影响

- 风险：
  1. FP8 路径依赖先前 PR：PR 依赖 #36061（FP8 activation-scheme 推断）与 #36060，若这两者的实现有边界情况（如动态 FP8 的 activation scale 缺失），可能引发加载失败或精度问题。
  2. 配置契约变更：set_comfy_layer_markers 现在接受 float8_e4m3fn，如果其他调用方（如非 MiniMax-H3 路径）传入了意外格式，原先的 ValueError 保护被放宽，可能掩盖错误配置。
  3. 调度顺序风险：在 get_quant_method 中 FP8 检查位于 INT8 之后、NVFP4 默认之前，若前缀同时命中多个配置（理论上不可能，但需注意 fp8_markers 与 int8_markers 的构造逻辑是否有重叠键），可能存在歧义。
  4. 测试覆盖有限：新增测试仅验证了单个 FP8 层的调度，未覆盖静态 FP8、动态 FP8 的权重加载细节，也未验证 ComfyFp8Config 内部逻辑在混合 NVFP4 场景下的端到端行为。
- 影响：
  1. 用户侧：MiniMax-H3 用户可以直接加载包含 NVFP4 + FP8 混合量化层的 Comfy 导出的社区检查点，无需手动指定 --quantization 或使用文件名规则，降低部署门槛。
  2. 系统侧：量化配置的分发逻辑（data-contract）扩展了容许的格式集合，影响所有走 ModelOptFp4Config 的模型加载路径；但影响面集中在 multimodal_gen 模块。
  3. 团队侧：该 PR 是 Diffusion 量化支持系列的一环（与 #36040、#36044、#36055 等连续演进），为后续更复杂的混合量化（如 W4A8、混合精度）铺路。 - 风险标记：依赖未合并的先前 PR，格式白名单放宽可能掩盖错误配置，测试未覆盖静态 / 动态 FP8 端到

端路径

关联脉络

- PR #36040 [diffusion] feat: support mixed w4a4 and int8 checkpoints: 同一功能线的前序工作，实现了 W4A4 + INT8 混合检查点的分发，本 PR 在其基础上新增 FP8 伴随层支持。
- PR #36044 [Diffusion] Load Comfy NVFP4 MiniMax H3 checkpoints: 实现了 Comfy NVFP4 MiniMax-H3 检查点的加载，本 PR 依赖其 NVFP4 解析与层标记机制。
- PR #36055 [Diffusion] Load MiniMax H3 GGUF text encoders: 同为 diffusion 量化体系扩展，涉及量化配置与加载器，与本 PR 共享相关模块（但功能分支不同）。