PR #36023 完整报告

sgl-project/sglang

[diffusion] Load serialized Comfy ConvRot INT8 native encoders

合并时间：2026-08-23 12:00

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/36023

执行摘要

- 一句话：支持原生加载 Comfy 序列化 INT8 编码器
- 推荐动作：值得精读。重点学习三点设计：一是 fail-closed 的加载准入（marker 未消费即报错，避免静默降级到不兼容的 Transformers 加载器）；二是 param_names_mapping 把 checkpoint 命名空间与 native 参数解耦的通用模式；三是 supports_input_partition 作为量化后端与 TP 分片之间的契约。这三个模式对后续把更多 Comfy/ 外部量化格式接入 SGLang 有直接参考价值。

功能与动机

PR body 的目标是 auto-detect and validate tensor-level Comfy FP8 / ConvRot INT8 markers for native encoder single-file overrides，并 keep quantized admission fail-closed so a native load error cannot silently fall back to an incompatible Transformers loader。作者检查了公共 Comfy-Org/MiniMax-H3/text_encoders/qwen3vl_32b_minimax_h3_int8_convrot.safetensors 的 header（未下载 payload）：350 个量化 language-linear 均为 INT8 权重、FP32 row scale，带 int8_tensorwise / ConvRot group-256 标记，而 visual tower 无 marker 保持源精度——这促使需要一条能直接消费该格式的原生加载路径，而不是依赖 Transformers 或绕道 Comfy 导出。

实现拆解

整个改造分 5 步推进：

1. 通用化 Comfy marker 解析器：把 minimax_h3_weights.py 中 inspect_minimax_h3_safetensors 内嵌的 marker 解析、dtype/shape 校验、缺失标记检查整体上提为 quantization_utils.inspect_comfy_quant_markers，并新增 param_name_mapper 参数支持 checkpoint 命名空间到 native 命名空间的映射；resolve_minimax_h3_checkpoint_quantization 也改为复用 resolve_comfy_checkpoint_quantization。minimax_h3_weights.py 从 98 行变更缩减为薄包装，H3 DiT 与文本编码器从此共用同一套校验逻辑。
2. 文本编码器加载链接入：text_encoder_loader._get_encoder_quant_config 新增 model_cls 参数，当常规配置与 safetensors 元数据都没有量化信息时，从 model_cls.param_names_mapping 构造 name_mapper（发现 stacked 映射的 merge_index 非 None 时直接拒绝），再调用 inspect_comfy_quant_markers 与 resolve_comfy_checkpoint_quantization 产出 KitchenInt8Config 或 ComfyFp8Config。

_require_quantized_encoder_layers 增加 quant_config 校验，要求 checkpoint 中每个 marker 前缀都被模型消费；load_customized 对量化加载中的其他异常统一包装为 ComponentCheckpointUnsupportedError，实现 fail-closed。权重迭代新增 include_checkpoint_weight，过滤 *.comfy_quant 元数据张量，避免把 JSON 标记当权重加载。

3. MiniMax-H3 Qwen3-VL 命名空间映射：minimax_h3_qwen3vl.py 声明 param_names_mapping（model.embed_tokens|layers|norm|rotary_emb.* → model.language_model.*、visual.* → model.visual.*、视觉 qkv → qkv_proj）并暴露为类属性，load_weights 与 should_materialize_checkpoint_weight 统一先做 _map_checkpoint_name；同时 qwen3vl.py 的 Qwen3VLModel / Qwen3VLTextModel 增加 prefix 参数并用 add_prefix 拼接，使 native 参数名与 Comfy checkpoint 命名空间对齐。
4. TP 分区兼容与序列化后处理：QuantizationConfig 基类新增 supports_input_partition 默认实现（返回 True）；KitchenInt8Config 按 marker 记录的 convrot_groupsize 校验 input_size_per_partition 是否整除，qwen3vl._make_text_row_linear 在 row-parallel 不满足该契约时退回复制权重——这正是 TP8 下 3200 宽 shard 会破坏 256 元素旋转组的应对方案。另外序列化 INT8 checkpoint 的 _process_quantized_encoder_weights 允许 process_device=None，即不做设备搬移直接后处理。
5. 测试与文档配套：test_text_encoder_loader.py 新增 3 个测试（Comfy 名称映射到 native 命名空间、INT8 权重文件配置出 KitchenInt8Config、未消费 marker 被拒绝）；test_transformer_quant.py 补充 supports_input_partition 断言；docs/docs/sglang-diffusion/quantization.mdx 补充通用 component-file CLI 与 Comfy 序列化 INT8 的说明。

关键文件：

- python/sglang/multimodal_gen/runtime/utils/quantization_utils.py（模块 量化检测；类别 source；类型 core-logic；符号 inspect_comfy_quant_markers, resolve_comfy_checkpoint_quantization）：新增通用 Comfy 量化 marker 解析与格式裁决函数，是文本编码器与 DiT 共享的核心工具。
- python/sglang/multimodal_gen/runtime/loader/component_loaders/text_encoder_loader.py（模块 编码器加载；类别 source；类型 dependency-wiring；符号 _get_encoder_quant_config, _require_quantized_encoder_layers, _process_quantized_encoder_weights, include_checkpoint_weight）：文本编码器加载链的核心接入点：marker 检测、fail-closed 校验、comfy_quant 键过滤、异常包装全部在这里生效。
- python/sglang/multimodal_gen/runtime/models/encoders/minimax_h3_qwen3vl.py（模块 权重映射；类别 source；类型 data-contract；符号 _map_checkpoint_name, _PARAM_NAMES_MAPPING）：声明 Comfy checkpoint 到 native 参数名的映射，是名称 canonicalize 与 marker 对齐的关键数据契约。
- python/sglang/multimodal_gen/runtime/loader/minimax_h3_weights.py（模块 H3 加载；类别 source；类型 refactor；符号 inspect_minimax_h3_safetensors, resolve_minimax_h3_checkpoint_quantization）：将原有 Comfy marker 解析删除并改为

复用通用工具，是本次重构收敛的直接体现。

- python/sglang/multimodal_gen/runtime/layers/quantization/configs/kitchen_int8_config.py（模块 量化后端；类别 source；类型 core-logic；符号 supports_input_partition）：新增 supports_input_partition，按 marker 的 ConvRot group size 校验 TP 输入分区，防止分片破坏旋转组。
- python/sglang/multimodal_gen/runtime/models/encoders/qwen3vl.py（模块 模型库；类别 source；类型 core-logic；符号 _make_text_row_linear, Qwen3VLModel, Qwen3VLTextModel）：row-parallel 决策纳入量化契约，并新增 prefix 参数以对齐 native 命名空间。
- python/sglang/multimodal_gen/runtime/layers/quantization/configs/base_config.py（模块 量化基类；类别 source；类型 core-logic；符号 supports_input_partition）：定义量化后端 TP 分区兼容性的默认契约，所有后端默认允许分区，特殊格式可覆盖。
- python/sglang/multimodal_gen/runtime/layers/quantization/comfy_fp8.py（模块 量化后端；类别 source；类型 core-logic；符号 ComfyFp8Config.selected）：新增强制 selected 列表，用于记录被消费的 marker 前缀，支撑 unconsumed 校验。
- python/sglang/multimodal_gen/test/unit/test_text_encoder_loader.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_comfy_language_checkpoint_name_maps_to_native_namespace, test_comfy_int8_weight_file_configures_native_encoder, test_rejects_unconsumed_comfy_marker）：新增 3 个核心测试，覆盖名称映射、INT8 文件配置与未消费 marker 拒绝，是 fail-closed 行为的直接保障。
- python/sglang/multimodal_gen/test/unit/test_transformer_quant.py（模块 单元测试；类别 test；类型 test-coverage）：补充 supports_input_partition 的整除语义断言，保护 TP 分区契约不被回归。
- docs/docs/sglang-diffusion/quantization.mdx（模块 文档；类别 docs；类型 documentation）：文档化通用 component-file CLI 与 Comfy 序列化 INT8 加载方式，是面向用户的功能说明。

关键符号：inspect_comfy_quant_markers, resolve_comfy_checkpoint_quantization, _get_encoder_quant_config, _require_quantized_encoder_layers, _process_quantized_encoder_weights, include_checkpoint_weight, _map_checkpoint_name, supports_input_partition

关键源码片段

python/sglang/multimodal_gen/runtime/loader/component_loaders/text_encoder_loader.py

文本编码器加载链的核心接入点：marker 检测、fail-closed 校验、comfy_quant 键过滤、异常包装全部在这里生效。

```
def _require_quantized_encoder_layers(
    model: nn.Module,
    component_name: str,
    quant_config: QuantizationConfig | None = None,
) -> None:
```

```
    # 声明量化的 checkpoint 必须真的构造出量化线性层，防止“加载了但没生效”。
    has_quantized_layers = any(
        isinstance(module, LinearBase)
        and module.quant_method is not None
        and not isinstance(module.quant_method, UnquantizedLinearMethod)
        for module in model.modules()
    )
    if not has_quantized_layers:
        raise ComponentCheckpointUnsupportedError(
            f"The native {type(model).__name__} implementation does not construct "
            f"quantized linear layers for {component_name!r}"
        )
    # Comfy 序列化格式要求 checkpoint 中每个 marker 前缀都被模型消费；
    # 未消费的 marker 说明 checkpoint 与 native 模型结构不匹配，必须显式失败。
    if isinstance(quant_config, (ComfyFp8Config, KitchenInt8Config)):
        missing = set(quant_config.layer_markers) - set(quant_config.selected)
        if missing:
            raise ComponentCheckpointUnsupportedError(
                f"The native {type(model).__name__} implementation did not consume "
                f"Comfy quantization markers for {component_name!r}: "
                f"{sorted(missing)[:5]}"
            )
```

python/sglang/multimodal_gen/runtime/models/encoders/minimax_h3_qwen3vl.py

声明 Comfy checkpoint 到 native 参数名的映射，是名称 canonicalize 与 marker 对齐的关键数据契约。

```
# Comfy 发布的 MiniMax-H3 encoder checkpoint 使用不带 language_model 前缀的命名空间，
# 本映射把 checkpoint 名称统一搬运到 native 参数命名空间。
_PARAM_NAMES_MAPPING = {
    r"^model\.(embed_tokens|layers|norm|rotary_emb)\.": r"model.language_model.\1.",
    r"^visual\.": r"model.visual.",
    r"^(model\.visual\.blocks\.\d+\.attn\.)qkv\.": r"\1qkv_proj.",
}
_MAP_CHECKPOINT_NAME = get_param_names_mapping(_PARAM_NAMES_MAPPING)


def _map_checkpoint_name(name: str) -> str:
    # get_param_names_mapping 返回（映射名，合并索引，是否合并）；encoder 场景
    # 不允许 stacked 合并，因此只取第一个返回值即可。
    return _MAP_CHECKPOINT_NAME(name)[0]


def load_weights(
    self,
    weights: Iterable[tuple[str, torch.Tensor]],
) -> set[str]:
    params = dict(self.named_parameters(remove_duplicate=False))
```

```
    loaded: set[str] = set()
    for name, loaded_weight in weights:
        # 权重名先映射到 native 命名空间，再走统一的消费 / 过滤逻辑。
        name = _map_checkpoint_name(name)
        if not self.should_materialize_checkpoint_weight(name):
            continue
        param_name = name
        param = params.get(param_name)
        if param is None:
            raise KeyError(
                "Unexpected MiniMax H3 Qwen3-VL checkpoint weight: "
                f"{name} (mapped to {param_name})"
            )
        # 后续走 weight_loader 并记录已加载参数，保证加载集合可审计。
        weight_loader = getattr(param, "weight_loader", default_weight_loader)
        try:
            can_keep_checkpoint_tensor = bool(
                getattr(self, "_keep_checkpoint_mapping", False)
                and weight_loader is default_weight_loader
                and param.device.type == "cpu"
                and loaded_weight.device.type == "cpu"
                and loaded_weight.dtype == param.dtype
                and tuple(loaded_weight.shape) == tuple(param.shape)
            )
            if can_keep_checkpoint_tensor:
                param.data = loaded_weight
            else:
                weight_loader(param, loaded_weight.to(param.dtype))
        except Exception as exc:
            raise RuntimeError(
                "Failed to load MiniMax H3 Qwen3-VL weight "
                f"{name!r}: checkpoint={tuple(loaded_weight.shape)}, "
                f"parameter={tuple(param.shape)}"
            ) from exc
        loaded.add(param_name)
    return loaded
```

评论区精华

本 PR 没有任何人工 review 评论或讨论线程，唯一的配套注释来自 mintlify[bot] 的文档 preview 部署（quantization.mdx 预览就绪）。作者 mickqian 同时担任提交者与合并者，单 commit 直接合入，设计权衡主要体现在 PR body 中显式声明的不变量：fail-closed 加载、marker 必须被消费、TP 分区不能破坏旋转组。

- 文档 preview 部署（无实质 review） (other): 无设计争议；作者 mickqian 同时为合并者，单 commit 完成合入。

风险与影响

- 风险：
  1. 共享解析器重构的回归面：minimax_h3_weights.py 删除 93 行重复实现，错误信息前缀从 MiniMax-H3 Comfy layer 变为 Comfy layer，依赖旧错误文案的外部脚本或测试可能受影响；行为等价性由 DiT 路径现有测试保障，但该 PR 本身未新增 DiT 侧回归用例。
  2. marker 检测的误判风险：inspect_comfy_quant_markers 会把任何 I8/F8 权重视为需要 comfy_quant 标记，非 Comfy 的序列化 INT8 checkpoint 若恰好缺标记会直接报错，属于可用性风险，但符合 fail-closed 设计。
  3. TP 分片保护的盲区：KitchenInt8Config.supports_input_partition 在 _serialized_group_sizes.get(prefix) 查不到时返回 True（允许分片），若 prefix 匹配失败可能跳过保护；TP8 下复制 3200 宽投影也会带来显存与加载时间上升。
  4. 验证缺口：作者声明未运行本地 GPU/runtime 测试；NVIDIA CI 通过，AMD ROCm CI 失败且未澄清原因；TP8 复制路径与 serialized INT8 的 device postprocess 路径缺少专门的数值 / 端到端测试。 - 影响：用户侧：MiniMax-H3 用户可直接使用 Comfy 发布的 INT8 单文件 encoder，显存约减半且保持 ConvRot 旋转组语义；其他带 Comfy FP8 marker 的 encoder 单文件覆盖也会自动获得检测与原生加载能力。系统侧：Comfy 序列化量化加载逻辑统一收敛到 quantization_utils，后续新增量化格式只需扩展 resolve_comfy_checkpoint_quantization。团队侧：文本编码器与 DiT 共享解析器，减少重复维护；supports_input_partition 成为新的量化后端契约，各后端需明确声明 TP 分区兼容性。 - 风险标记：核心加载路径变更，共享解析器重构，TP8 复制路径缺测试，AMD CI 失败未澄清，marker 检测可能误判

关联脉络

- PR #36060 [Diffusion] Infer Comfy FP8 activation scaling: 同一条功能线：共享 quantization_utils 与 comfy_fp8 的 Comfy 量化标记解析与配置推断，本 PR 的 resolve_comfy_checkpoint_quantization 是对该路径的推广。
- PR #36076 [Diffusion] Support compact Qwen3-VL conditioning for MiniMax H3: 改动了相同的 minimax_h3_qwen3vl.py 与 text_encoder_loader.py 文件，属于同一 MiniMax-H3 编码器系列的功能演进。
- PR #36067 [Diffusion] Load Diffusers MiniMax H3 components natively: H3 DiT 原生加载是本 PR Comfy marker 解析逻辑的来源，本 PR 将其上提为共享工具并让文本编码器复用。