PR #36046 完整报告

sgl-project/sglang

[Diffusion] Load Comfy NVFP4-AWQ text encoders

合并时间：2026-08-25 15:43

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/36046

执行摘要

- 一句话：支持自动加载 Comfy NVFP4-AWQ 文本编码器
- 推荐动作：值得精读，尤其是想理解“保留量化存储、按需反量化”这一设计权衡的读者。建议关注三个点：一是 inspect_comfy_quant_markers 如何用逐层 marker 验证替代显式量化参数；二是 ComfyFullPrecisionNvfp4LinearMethod 故意跳过 process_weights_after_loading 以保持可移植性的取舍；三是文档明确不承诺 FP4 加速的诚实表述方式。对扩展其他 Comfy 量化格式的开发者，本 PR 的标记解析与分派模式可以直接借鉴。

功能与动机

PR body 明确说明目标是“auto-detect Comfy NVFP4-AWQ encoder checkpoints from their per-layer metadata”，并希望“preserve the official row-wise INT8 embedding and packed NVFP4 storage”以及“honor AWQ input pre-scales, Comfy's high-nibble-first packing, and swizzled block scales”。同时刻意“dequantize only the active matrix for BF16/FP16 compute, without advertising native FP4 acceleration”，避免对用户承诺不存在的性能收益。该 PR 基于 #36044 引入的共享 Comfy NVFP4 检查点布局，使扩散文本编码器不需要用户指定量化参数即可工作。

实现拆解

实现按以下 5 个步骤拆解：

1. 扩展检查点标记解析（quantization_utils.py）：在 inspect_comfy_quant_markers 中新增 nvfp4 格式支持：把 U8 权重加入可标记 dtype；要求 weight_scale_2 为标量 F32、weight_scale 为 FP8 E4M3、权重二维且 weight_shape[1]*2 能被 16 整除；校验 pre_quant_scale 的 dtype 与形状，并在 marker 中写入 _has_pre_quant_scale；同时为 INT8 tensorwise 标记增加 _is_rowwise，为加载器分派提供依据。这一步是整个自动检测的入口，任何形状或 dtype 不符都会直接报错，避免带病加载。
2. 新增 Comfy NVFP4 量化方法（新增 comfy_nvfp4.py）：新增 _register_parameter 统一登记参数与并行维度；ComfyRowwiseInt8EmbeddingMethod 把 embedding 保存为 INT8 权重加逐行 FP32 scale，apply 直接抛 NotImplementedError，仅支持通过 embedding() 对命中行做 gather 后乘 scale 的按需反量化；ComfyFullPrecisionNvfp4LinearMethod 继承 ModelOptFp4LinearMethod，在 create_weights 中保留 U8 打包权重、FP8 块 scale、标量 weight_scale_2 和可选

pre_quant_scale，process_weights_after_loading 为空，明确放弃 ModelOpt 的 Blackwell 专用内核预处理，以便直接消费序列化表示。

3. 接入文本编码器加载器（text_encoder_loader.py）：_configure_encoder_quantization 相关分派逻辑识别 ComfyNvfp4Config；_require_quantized_encoder_layers 的量化配置元组加入 ComfyNvfp4Config，确保所有序列化 layer marker 都必须被模型真实消费，否则抛 ComponentCheckpointUnsupportedError；load_model 的权重后处理设备分支把 ComfyNvfp4Config 与 QuantoInt8Config 归为一类，保持原始序列化布局不做CPU卸载。
4. 扩展共享反量化器（srt/layers/quantization/dequantization.py）：dequantize_nvfp4 新增 high_nibble_first: bool = False 参数，默认保持 ModelOpt/Quark 的低半字节对应偶数索引布局；Comfy 导出为高半字节优先，通过 (high, low) 交换解包顺序。原有调用方全部使用默认值，行为不变。
5. 测试与文档配套：test_text_encoder_loader.py 新增两个单元测试，分别覆盖 safetensors 元数据到 embedding/linear marker 的映射，以及带 pre_quant_scale 的 portable Linear 与 row-wise embedding 的前向输出；文档在 quantization.mdx 的对照表中增加 comfy-nvfp4-full-precision 行，并在 MiniMax-H3 cookbook 中说明 --component-paths.text_encoder 的无标志覆盖用法，明确该路径是“memory path rather than a native FP4 speed path”。

关键文件：

- python/sglang/multimodal_gen/runtime/layers/quantization/comfy_nvfp4.py（模块 量化方法；类别 source；类型 core-logic；符号 _register_parameter, ComfyRowwiseInt8EmbeddingMethod, create_weights, apply）：新增文件，实现 Comfy NVFP4-AWQ 编码器的两种量化方法：行式 INT8 embedding 按需反量化与完整精度 NVFP4 Linear，是本 PR 的核心实现。
- python/sglang/multimodal_gen/runtime/utils/quantization_utils.py（模块 检查点解析；类别 source；类型 core-logic）：扩展检查点标记解析逻辑，新增 nvfp4 格式的完整形状校验与 pre_quant_scale 检测，是自动分派的前提。
- 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_nvfp4_awq_weight_file_maps_embedding_and_linear_markers, test_nfp4_awq_portable_linear_and_rowwise_embedding）：新增两个核心单元测试，覆盖 NVFP4-AWQ 检查点元数据映射与 portable 反量化前向结果，是保证自动检测正确性的主要依据。
- python/sglang/multimodal_gen/runtime/loader/component_loaders/text_encoder_loader.py（模块 加载器；类别 source；类型 dependency-wiring）：将 ComfyNvfp4Config 接入文本编码器加载器的量化分派与层消费校验，决定新路径何时生效。
- python/sglang/srt/layers/quantization/dequantization.py（模块 反量化；类别 source；类型 core-logic；符号 dequantize_nvfp4）：共享反量化函数 dequantize_nvfp4 新增 high_nibble_first 参数，用于区分 Comfy 与 ModelOpt 的 NVFP4 半字节打包顺序，影响所有 NVFP4 反量化调用方。
- docs/docs/sglang-diffusion/quantization.mdx（模块 文档；类别 other；类型 documentation）：量化支持矩阵新增 comfy-nvfp4-full-precision 条目，明确该路径是内存优化而非 FP4 原生加速，防止用户误用。

- docs/cookbook/diffusion/MiniMax/MiniMax-H3.mdx（模块 文档；类别 other；类型 documentation）：MiniMax-H3 cookbook 补充无标志的 --component-paths.text_encoder 覆盖用法，为用户提供可复制的部署示例。

关键符号：dequantize_nvfp4, inspect_comfy_quant_markers, _register_parameter, ComfyRowwiseInt8EmbeddingMethod.embedding, ComfyRowwiseInt8EmbeddingMethod.create_weights, ComfyFullPrecisionNvfp4LinearMethod.create_weights, ComfyFullPrecisionNvfp4LinearMethod.process_weights_after_loading, _require_quantized_encoder_layers

关键源码片段

python/sglang/srt/layers/quantization/dequantization.py

共享反量化函数 dequantize_nvfp4 新增 high_nibble_first 参数，用于区分 Comfy 与 ModelOpt 的 NVFP4 半字节打包顺序，影响所有 NVFP4 反量化调用方。

```
def dequantize_nvfp4(
    w_q: torch.Tensor,
    w_s: torch.Tensor,
    w_s2: Optional[torch.Tensor],
    out_dtype: torch.dtype = torch.bfloat16,
    high_nibble_first: bool = False,
) -> torch.Tensor:
    """NVFP4 -> out_dtype。

    w_q: uint8 [..., out, in/2] 打包 E2M1；
    默认（ModelOpt/Quark）低半字节对应偶数索引；
    Comfy 导出则相反，需要传 high_nibble_first=True。
    w_s: FP8 E4M3 [..., out, in/16] 逐块 scale；
    w_s2: 可选 FP32 标量，整体乘到逐块 scale 上。
    """
    ... # batch/out_dim/in_dim 计算与 E2M1 LUT 准备保持不变
    low = (w_q & 0xF).to(torch.int64) # 低 4 位索引
    high = (w_q >> 4).to(torch.int64) # 高 4 位索引
    # Comfy 高半字节优先时，把高半字节放到偶数索引位置
    first, second = (high, low) if high_nibble_first else (low, high)
    deq = torch.empty(*batch, out_dim, in_dim, dtype=torch.float32, device=device)
    deq[..., 0::2] = lut[first]
    deq[..., 1::2] = lut[second]
    scale = w_s.to(torch.float32)
    if w_s2 is not None:
        scale = scale * w_s2.to(torch.float32)
    ... # 乘 scale 并转到 out_dtype
```

评论区精华

该 PR 没有任何 review 评论或 GitHub 讨论线程，设计决策主要体现在 23 个提交的演进中：从 W4A8、W4A4 逐步扩展到 NVFP4 H3 DiT，再到本 PR 的 NVFP4-AWQ 文本编码器；其中“Preserve standard ModelOpt NVFP4 layout”与“Avoid overstating H3 NVFP4 validation”两个提交反映了实现中途曾对齐 ModelOpt 标准布局，并主动收敛文档措辞；多个 “merge encoder quant marker fix” 提交则说明编码器 marker 映射在 CI 中经历过反复修正。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 共享反量化器风险：dequantize_nvfp4 被 SRT 的 ModelOpt/Quark 路径共用，新增 high_nibble_first 默认值为 False，保持旧行为；但如果未来其他 Comfy 检查点接入时忘记传 True，会静默产生错误数值而不是报错。
  2. 自动检测风险：inspect_comfy_quant_markers 对 nvfp4 标记做了严格形状校验，一旦 Comfy 官方导出版本变化（如 scale 形状或 pre_quant_scale dtype 调整），加载会直接失败；没有提供显式开关绕过校验。
  3. 功能边界风险：ComfyRowwiseInt8EmbeddingMethod.apply 直接抛异常，任何误将该 embedding 当作 Linear 调用的路径都会崩溃；文档已说明仅支持 lookup。
  4. 性能预期风险：PR 明确定位为内存路径而非原生 FP4 加速，但用户看到 NVFP4 标签可能误以为有 Blackwell 加速，需要依赖文档纠正。
  5. 验证覆盖风险：运行时验证仅依赖 NVIDIA CI，AMD/XPU 等后端没有覆盖此加载路径。
     - 影响：用户侧：MiniMax-H3 等扩散工作流现在可以直接用 --component-paths.text_encoder 指定 Comfy NVFP4-AWQ 编码器 safetensors，无需手动挑选量化选项，模型加载内存显著下降。系统侧：为 sglang.multimodal_gen 的组件量化分派增加了一个新的自动检测类别，并将检查点格式约定（高半字节优先、swizzled scale、AWQ 预缩放）固化在共享工具中，后续其他扩散模型可复用同一套机制。团队侧：该 PR 是“Comfy 检查点家族”支撑的最后一块拼图，降低了维护多套量化装载逻辑的负担，但也在共享 dequantize_nvfp4 上新增了一个需要所有调用方理解的分支。
     - 风险标记：共享反量化器新增打包顺序分支，自动检测缺少显式开关，运行时验证依赖 NVIDIA CI，无原生 FP4 加速需依赖文档纠偏

关联脉络

- PR #36061 [Diffusion] Dispatch mixed Comfy NVFP4 and INT8 layers: 同一 Comfy NVFP4 支持线，处理 MiniMax-H3 DiT 的逐层 NVFP4/INT8 分派，与本 PR 共用同一检查点布局假设。
- PR #36066 [diffusion] feat: dispatch fp8 companions in mixed nvfp4 checkpoints: 延续混合 NVFP4 检查点的量化分派能力，与本 PR 同属 diffusion 量化自动检测演进方向。
- PR #36035 [Diffusion] Add component-scoped quantization overrides: 引入组件级量化覆盖与 --component-paths 机制，本 PR 的文本编码器无标志覆盖正是建立在该机制之上。