PR #29903 完整报告

sgl-project/sglang

[diffusion] fix: fix z-Image online fp8 quantization crash with dit_cpu_offload

合并时间：2026-07-04 23:40

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/29903

执行摘要

- 一句话：修复在线 FP8 量化与 DiT CPU offload 冲突导致崩溃
- 推荐动作：推荐所有人阅读，尤其是 diffusion 反量化加载路径的维护者。此 PR 展示了一个由隐式回退引发的隐蔽 bug 及其系统化修复方法： 1) 在加载阶段对 dit_cpu_offload 进行约束，2) 移除静默回退，3) 在调用链末端增加防御性检查。三个层次环环相扣，体现了防御性工程实践。

功能与动机

关联 Issue #29833 报告：使用 sglang generate --model-path Tongyi-MAI/Z-Image-Turbo --quantization fp8 时发生 ValueError: Cannot find any of ['quant_method'] 并最终导致 OOM。根本原因是 dit_cpu_offload 将模型权重移至 CPU，而在线 FP8 量化内核只能处理 CUDA/NPU张量；加载器在量化失败后未报错而是回退到未量化的BF16模型，引发后续崩溃。

实现拆解

1. 扩展 dit_cpu_offload 禁用逻辑：在 transformer_load_utils.py 中，将原来仅针对 modelopt_fp8 的 _maybe_disable_incompatible_dit_offload_modes 修改为对所有在线量化（fp8、mxfp4 等）均强制禁用 dit_cpu_offload，确保权重在加载时位于设备上。
2. 阻止静默回退：在 transformer_loader.py 的 should_raise_customized_load_error 中增加条件：当指定了 quantization 时，也要求加载器抛出错误而不是回退到原生 diffusers 模型，避免隐藏真实问题。
3. 添加防御性检查：在 zimage.py 的 get_freqs_cis 方法开头增加对 rotary_emb is None 的判断，若为 None 则抛出带有明确指引的 ValueError，帮助用户定位加载日志中的真实错误。
4. 测试配套：新增 test_modelopt_fp8_adapter_does_not_change_online_fp8_offload 测试验证在线 FP8 时 dit_cpu_offload 不被修改；新增 test_online_fp8_requires_device_weight_processing 测试验证 _requires_device_weight_processing 对在线 FP8 返回 True、对预量化 FP8 返回 False。

关键文件：

- python/sglang/multimodal_gen/runtime/loader/transformer_load_utils.py（模块 加载器；类别 source；类型 core-logic；符号 _requires_device_weight_processing, TransformerQuantLoadSpec, _ModelOptFp8OffloadAdapter）：核心改动：新增

requires_device_weight_processing 字段和 _requires_device_weight_processing 函数，标识在线量化需要设备端处理权重；同时修改 _ModelOptFp8OffloadAdapter._maybe_disable_incompatible_dit_offload_modes 使其不再对在线 FP8 禁用 dit_cpu_offload（因为该适配器只应处理预量化 modelopt_fp8，而在线 FP8 需要保留 cpu offload 以便在 fsdp_load 中延迟 offload）。

- python/sglang/multimodal_gen/runtime/loader/fsdp_load.py（模块 加载器；类别 source；类型 core-logic；符号 maybe_load_fsdp_model）：新增 defer_cpu_offload_until_after_weight_processing 参数，在在线量化场景下先加载权重到设备、执行 post_load_weights 后再 CPU offload，确保量化内核在设备权重上执行。
- python/sglang/multimodal_gen/runtime/loader/component_loaders/transformer_loader.py（模块 加载器；类别 source；类型 core-logic；符号 should_raise_customized_load_error, load_customized）：修改 should_raise_customized_load_error 强制量化加载失败时报错；在 load_customized 中传递 defer_cpu_offload_until_after_weight_processing 标志。
- python/sglang/multimodal_gen/configs/pipeline_configs/zimage.py（模块 配置；类别 source；类型 core-logic；符号 get_freqs_cis）：在 get_freqs_cis 中添加防御性检查，当 rotary_emb 为 None 时抛出明确错误，帮助排查因加载回退导致的问题。
- python/sglang/multimodal_gen/test/unit/test_layerwise_offload.py（模块 测试；类别 test；类型 test-coverage；符号 test_modelopt_fp8_adapter_does_not_change_online_fp8_offload）：新增测试 test_modelopt_fp8_adapter_does_not_change_online_fp8_offload，验证在线 FP8 量化时 dit_cpu_offload 不被 _ModelOptFp8OffloadAdapter 修改（因为该适配器只用于预量化 modelopt_fp8）。
- python/sglang/multimodal_gen/test/unit/test_transformer_quant.py（模块 测试；类别 test；类型 test-coverage；符号 test_online_fp8_requires_device_weight_processing）：新增测试 test_online_fp8_requires_device_weight_processing，验证 _requires_device_weight_processing 对在线 FP8 返回 True、对预量化 checkpoint FP8 返回 False。

关键符号：_requires_device_weight_processing, maybe_load_fsdp_model, should_raise_customized_load_error, get_freqs_cis, _ModelOptFp8OffloadAdapter._maybe_disable_incompatible_dit_offload_modes

关键源码片段

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

核心改动：新增 requires_device_weight_processing 字段和 _requires_device_weight_processing 函数，标识在线量化需要设备端处理权重；同时修改 _ModelOptFp8OffloadAdapter._maybe_disable_incompatible_dit_offload_modes 使其不再对在线 FP8 禁用 dit_cpu_offload（因为该适配器只应处理预量化 modelopt_fp8，而在线 FP8 需要保留 cpu offload 以便在 fsdp_load 中延迟 offload）。

```
@dataclass
class TransformerQuantLoadSpec:
    """Resolved loading plan for a transformer checkpoint."""
```

```
    safetensors_list: list[str]
    quant_config: Optional[QuantizationConfig]
    nunchaku_config: Optional[NunchakuConfig]
    param_dtype: Optional[torch.dtype]
    # 新增：是否需要权重在设备上处理（在线量化需要 CUDA/NPU 内核）
    requires_device_weight_processing: bool = False
    post_load_hooks: list[PostLoadHook] = field(default_factory=list)

    @property
    def runtime_quant_config(self) -> Optional[object]:
        if self.quant_config is not None:
            return self.quant_config
        return self.nunchaku_config


def _requires_device_weight_processing(
    quant_config: Optional[QuantizationConfig],
) -> bool:
    """Return whether post-load weight processing needs CUDA/NPU tensors."""
    quant_name = _get_quant_config_name(quant_config)
    if quant_name == "fp8":
        # 在线 FP8 需要设备上处理；预量化 checkpoint 已经序列化，不需要
        return not getattr(quant_config, "is_checkpoint_fp8_serialized", False)
    if quant_name == "mxfp4":
        return not getattr(quant_config, "is_checkpoint_mxfp4_serialized", False)
    return False
```

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

新增 defer_cpu_offload_until_after_weight_processing 参数，在在线量化场景下先加载权重到设备、执行 post_load_weights 后再 CPU offload，确保量化内核在设备权重上执行。

```
def maybe_load_fsdp_model(
    ...
    defer_cpu_offload_until_after_weight_processing: bool = False,
) -> torch.nn.Module:
    """
    ...
    defer_cpu_offload_until_after_weight_processing: If True, keep weights
        on device until process_weights_after_loading completes, then apply
        non-FSDP CPU offload.
    """
    ...
    defer_cpu_offload = bool(
        cpu_offload and defer_cpu_offload_until_after_weight_processing
    )
    if defer_cpu_offload and use_fsdp:
        logger.warning(...)
        defer_cpu_offload = False
    load_cpu_offload = cpu_offload and not defer_cpu_offload
```

```
# 加载时使用 load_cpu_offload 控制
load_model_from_full_model_state_dict(
    model,
    weight_iterator,
    device,
    param_dtype,
    strict=strict,
    cpu_offload=load_cpu_offload,
    ...
)
... # 执行 post_load_weights 等
if defer_cpu_offload:
    model.to("cpu") # 权重处理完成后才 offload
...
```

评论区精华

本 PR 无 review 评论。设计决策隐含在 PR body 与代码中：dit_cpu_offload 应被在线量化禁用，而 dit_layerwise_offload 保留，（因为逐层 offload 在运行时恢复 FP8 张量 stride 后仍能生效）；量化失败时倾向于快速失败（fail-fast）而不是静默回退以避免难调试的后续崩溃。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. GPU 内存增加：禁用 dit_cpu_offload 会导致 DiT 权重常驻 GPU，对内存敏感场景可能增大 OOM 风险，但用户仍可使用 --dit-layerwise-offload 逐层 offload。
  2. 加载路径变更：失败时抛出异常可能中断原本部分可用的回退路径（例如某些未完全兼容的模型），但这恰恰是期望的行为——让问题显式暴露。
  3. 影响范围：仅影响使用在线量化的 diffusion 模型（Z-Image 及其他调用此加载路径的模型），不影响预量化 checkpoint 或非 diffusion 场景。
  4. 测试覆盖：单测覆盖了新逻辑，但缺少端到端集成测试验证完整的加载 + 推理流程。 - 影响：用户：修复了 Z-Image 在线 FP8 量化的崩溃问题，使此已验证路径可用。使用在线量化的用户可获更稳定的体验。系统：增加了一个显式的 requires_device_weight_processing 标志位，扩展了 TransformerQuantLoadSpec 和 maybe_load_fsdp_model 的接口，未来可按需扩展。团队：明确了在线量化与 offload 策略的交互约定，减少了类似问题的调试难度。
- 风险标记：可能增加 GPU 内存占用，移除静默回退可能暴露新模型兼容性问题

关联脉络

- PR #30110 [diffusion] fix: shut down diffusion workers on serve exit: 同属 diffusion 模块基础设施修复，但功能无直接关联。