PR #29742 完整报告

sgl-project/sglang

[diffusion] Fix Z-Image accuracy

合并时间：2026-07-08 09:08

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/29742

执行摘要

- 一句话：修复动态批处理下 Z-Image 的精度问题
- 推荐动作：本 PR 修复了一个影响用户体验的关键精度 bug，且实现细节（保持 bf16 精度的 RMSNorm、Triton 融合内核的 fallback 机制、基于主机端 ranges 的 varlen 掩码构建）具有通用参考价值。关注扩散模型批处理的设计者值得精读。建议在后续 PR 中考虑将 batched freqs_cis 缓存以进一步优化推理吞吐。

功能与动机

关联 Issue #28502 报告了在使用 --batching-mode dynamic 时，Z-Image-Turbo 生成图像出现线条扭曲、主体缺失等严重降质问题。进一步分析发现，当前实现存在三个核心差异：

1) Qwen3 编码器默认 position_ids 未适配批次维度，导致 RoPE 计算错误； 2) 使用调度器默认 sigma 路径和外部 autocast，与官方 bf16 全精度轨迹不一致； 3) 批处理时未正确设置 RoPE 偏移和注意力掩码，使得不同请求的 token 相互干扰。

实现拆解

1. 自定义 RMSNorm 保持 bf16 精度 - 在 python/sglang/multimodal_gen/runtime/models/dits/zimage.py 中新增 ZImageRMSNorm 类，其 forward 全程在 bf16 域内计算，不与原生实现中共享的 RMSNorm（强制 fp32 累加）共用归一化轨迹。 - 同时新增 zimage_rmsnorm_tanh_mul_add 和 zimage_rmsnorm_scale 函数，优先尝试 Triton 融合内核，否则回退到纯 Python 实现。
2. Triton 融合内核 - 新增文件 python/sglang/jit_kernel/diffusion/triton/zimage_native_norm.py，实现 _rmsnorm_scale_kernel 和 _rmsnorm_tanh_residual_kernel，将归一化、缩放、tanh 门控和残差加法合并在单个内核中，避免中间张量创建和精度转换。 - 辅助函数 _can_use 检查张量是否连续、bf16、CUDA 且维度 ≤8192，确保内核安全执行。
3. 修复注意力层 QK-Norm - 在 ZImageAttention.__init__ 中将 self.norm_q 和 self.norm_k 从通用 RMSNorm 替换为 ZImageRMSNorm，并通过 enable_zimage_qk_fusion 标志控制（仅 quant_config is None 时启用）。 - forward 接口新增 rope_cos_sin_cache、rope_positions、attn_mask 和 attn_mask_meta 参数，支持传递批处理 RoPE 缓存和变长注意力掩码元数据。
4. 批处理 RoPE 与变长注意力掩码 - 在 zimage.py 中添加 _build_single_sample_freqs_cis、_pad_freqs_cis_to_length 和 _build_batched_freqs_cis 方法，为每张图片和字幕独立计

算 RoPE 位置，沿序列维度拼接并填充到目标长度。 - 在 python/sglang/multimodal_gen/runtime/layers/attention/layer.py 中新增 build_varlen_mask_meta_from_lengths 和 build_varlen_mask_meta_from_ranges，从主机端长度 / 区间构建 cu_seqlens、indices、inv_indices 等元数据，避免 GPU nonzero 动态形状路径。

5. Pipeline 配置与测试配套 - python/sglang/multimodal_gen/configs/pipeline_configs/zimage.py：覆盖 prepare_sigmas 使用显式 flow matching sigma 调度，禁用 enable_autocast。 - 新增 test_qwen3_encoder.py 测试 position_ids 批次形状；扩展 test_zimage_pipeline_config.py 覆盖新归一化公式、sigma 调度、批处理 RoPE 偏移；更新 test_consistency_metrics.py 为 GT 文件添加 h100/ 平台前缀；更新 perf_baselines/h100.json。 所有测试均以单 GPU 模式通过。

关键文件：

- python/sglang/multimodal_gen/runtime/models/dits/zimage.py（模块 图像模型；类别 source；类型 data-contract；符号 ZImageRMSNorm, init, forward, zimage_rmsnorm_tanh_mul_add）：核心变更文件，实现 ZImageRMSNorm、批处理 RoPE 构建和注意力层适配
- python/sglang/jit_kernel/diffusion/triton/zimage_native_norm.py（模块 JIT 内核；类别 source；类型 core-logic；符号 _tanh, _rmsnorm_scale_kernel, _rmsnorm_tanh_residual_kernel, _flat_row_stride）：新增 Triton 融合内核，提供 bf16 域内的 RMSNorm 缩放和 tanh 残差运算
- python/sglang/multimodal_gen/runtime/layers/attention/layer.py（模块 注意力层；类别 source；类型 dependency-wiring；符号 build_varlen_mask_meta_from_lengths, build_varlen_mask_meta_from_ranges）：新增 build_varlen_mask_meta_from_lengths 和 build_varlen_mask_meta_from_ranges 函数，支持主机端长度驱动变长掩码构建
- python/sglang/multimodal_gen/test/unit/test_qwen3_encoder.py（模块 编码器测试；类别 test；类型 test-coverage；符号 _CaptureLayer, init, forward, _IdentityNorm）：新增单元测试验证 Qwen3 编码器 position_ids 批次形状
- python/sglang/multimodal_gen/test/unit/test_zimage_pipeline_config.py（模块 管道配置测试；类别 test；类型 test-coverage；符号 test_rmsnorm_native_formula, test_explicit_sigmas, test_autocast_disabled, test_zimage_negative_prompt_rotary_embeddings_use_negative_prompt_len）：扩展测试覆盖 RMSNorm 公式、显式 sigma 调度、批处理 RoPE 偏移

关键符号：ZImageRMSNorm.init, ZImageRMSNorm.forward, zimage_rmsnorm_tanh_mul_add, zimage_rmsnorm_scale, _build_single_sample_freqs_cis, _build_batched_freqs_cis, build_varlen_mask_meta_from_lengths, build_varlen_mask_meta_from_ranges, _rmsnorm_scale_kernel, _rmsnorm_tanh_residual_kernel, test_default_position_ids_batch_shape, test_rmsnorm_native_formula, prepare_sigmas

关键源码片段

python/sglang/jit_kernel/diffusion/triton/zimage_native_norm.py

新增 Triton 融合内核，提供 bf16 域内的 RMSNorm 缩放和 tanh 残差运算

```
@triton.jit
def _rmsnorm_scale_kernel(
    y_ptr, x_ptr, weight_ptr, scale_ptr,
    x_row_stride, scale_row_stride, seq_len,
    dim: tl.constexpr, eps: tl.constexpr, block_dim: tl.constexpr,
):
    row = tl.program_id(0)
    offsets = tl.arange(0, block_dim)
    mask = offsets < dim

    x = tl.load(x_ptr + row * x_row_stride + offsets, mask=mask, other=0.0)
    # 在 bf16 中计算方差并求 rstd
    square = (x * x).to(tl.bfloat16)
    mean_square = (tl.sum(square, axis=0) / dim).to(tl.bfloat16)
    rstd = tl.rsqrt((mean_square + eps).to(tl.bfloat16).to(tl.float32)).to(tl.bfloat16)

    batch = row // seq_len
    weight = tl.load(weight_ptr + offsets, mask=mask, other=0.0)
    scale = tl.load(scale_ptr + batch * scale_row_stride + offsets, mask=mask, other=0.0)
    # 归一化 -> 加权 -> 缩放，全程 bf16
    y = (((x * rstd).to(tl.bfloat16) * weight).to(tl.bfloat16) * scale).to(tl.bfloat16)
    tl.store(y_ptr + row * dim + offsets, y, mask=mask)


def zimage_rmsnorm_scale(
    x: torch.Tensor, weight: torch.Tensor, scale: torch.Tensor, eps: float,
) -> torch.Tensor | None:
    # 检查所有输入都是 CUDA、bf16、连续且维度兼容
    if not _can_use(x, weight, scale):
        return None # 条件不满足时回退到 Python 实现
    dim = x.shape[-1]
    x_rows = x.numel() // dim
    scale_rows = scale.numel() // dim
    if x_rows % scale_rows != 0:
        return None
    seq_len = x_rows // scale_rows
    y = torch.empty_like(x, memory_format=torch.contiguous_format)
    with torch.get_device_module().device(x.device):
        _rmsnorm_scale_kernel[(x_rows,)](
            y.reshape(-1, dim), x, weight, scale,
            _flat_row_stride(x), _flat_row_stride(scale), seq_len,
            dim, eps, block_dim=triton.next_power_of_2(dim), num_warps=8,
        )
    return y
```

评论区精华

> gemini-code-assist[bot]建议在 forward 中缓存 batched freqs_cis 以避免跨 denoising step 的冗余计算，认为形状和设备在 step 间不变，可以基于输入 hash 缓存。目前 PR 未采纳该建议。
>
> mickqian询问问题是否仅出现在 Z-Image 以及是否需要更新 ground truth 图片。作者确认仅 Z-Image 受影响，ground truth 与官方一致无需更新。
>
> mickqian对一处 torch.stack 场景建议使用 torch.as_tensor 以提高效率，该评论未收到直接回复。

- 缓存 batched freqs_cis 避免跨步重复计算 (performance): 当前 PR 未采纳此建议，batched freqs_cis 在每个 forward 中重新计算。
- 使用 torch.as_tensor 替代 torch.stack (style): 未明确反馈是否采纳。

风险与影响

- 风险：
  1. 回归风险：注意力层中 QK norm 从通用 RMSNorm 切换到 ZImageRMSNorm，仅在 quant_config is None 时生效，不影响已启用量化的模型。新增的变长注意力元数据函数仅被 Z-Image 调用，但可能对依赖 build_varlen_mask_meta 的其他扩散模型造成影响，需确认兼容性。
  2. 性能风险：批处理 RoPE 构建在每次 forward 中重复计算，对于极长序列可能引入额外延迟；但从基准测试看，由于融合内核的加速，总体单步时间未增加甚至略微下降。
  3. 配置兼容性：enable_autocast 默认关闭，显式 sigma 调度覆盖基类配置，依赖 autocast 或默认 sigma 的模型或需要调整 pipeline 配置。
  4. 数据一致性与 CI：一致性基准测试文件路径改为 h100/ 前缀，需要确保 CI 数据仓库（ci-data）中存在对应文件，否则测试将失败。
- 影响：
  - 用户：Z-Image 动态批处理场景下图像质量与官方实现对齐，种子不变时输出稳定；单张生成速度提升约 1.4%，批处理 (size=5) 速度提升约 2.6%。
  - 系统：新增一个 Triton 内核文件 zimage_native_norm.py，无额外运行时依赖；变长注意力元数据函数可复用。
  - 团队：Z-Image 一致性 GT 图片需重新生成并上传至 ci-data 的 diffusion-ci/consistency_gt/official_generated/ 和 sglang_generated/，并在 _gt_case_registrations.py 中注册新 GT 版本。
  - 风险标记：批处理填充 mask 变更可能影响其他扩散模型，新增 Triton 内核需要 CUDA 兼容性，配置默认值更改 (autocast 关闭) 可能影响用户期望，一致性 GT 基准需要 CI 数据仓库同步

关联脉络

- 暂无明显关联 PR