PR #24313 完整报告

sgl-project/sglang

[diffusion] chore: align LTX-2 with official

合并时间：2026-05-07 08:46

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24313

执行摘要

- 一句话：对齐 LTX-2 与官方实现的注意力语义和数值精度
- 推荐动作：建议精读以下部分：
  - Gemma3 注意力掩码和 GQA 处理方式的变更（gemma_3.py）
  - NumPy 双精度 RoPE 频率计算的实现（ltx_2.py / ltx_2_connector.py）
  - res2s 标量精度对齐策略（ltx_2_denoising.py）
  - 组件级注意力后端自动配置（server_args.py）

这些变更体现了将非标准注意力路径与官方逐位对齐的典型方法，值得扩散模型开发者参考。

功能与动机

Align native LTX text-encoder attention behavior with the official implementation while preserving high-performance attention backends outside the text encoder path. Keep CI consistency gates honest by using official GT only for cases whose request semantics are currently comparable.

实现拆解

1. Gemma3 text encoder 注意力对齐：在 python/sglang/multimodal_gen/runtime/models/encoders/gemma_3.py 中，将 attention mask 从 additive bf16 mask 改为 bool keep-mask，并显式 repeat K/V 处理 GQA 而非依赖 enable_gqa=True；RoPE 计算从预计算 buffer 改为设备端实时生成，以匹配 LTX 方式。
2. RoPE 频率计算对齐：在 python/sglang/multimodal_gen/runtime/models/dits/ltx_2.py 和 python/sglang/multimodal_gen/runtime/models/adapter/ltx_2_connector.py 中新增 _ltx2_rope_freq_grid_np 和 _ltx2_connector_rope_freq_grid_np 函数，利用 NumPy float64 生成双精度频率网格，并使用 functools.lru_cache 缓存结果；在 double_precision 分支中替换原有 torch 计算路径。
3. res2s 调度标量精度对齐：在 python/sglang/multimodal_gen/runtime/pipelines_core/stages/ltx_2_denoising.py 中新增 _ltx2_phi_scalar、_ltx2_get_res2s_coefficients_scalar 和 _ltx2_res2s_step_size_scalar 函数，以标量精度计算 SDE 系数，避免张量运算中的精度损失；同时调整 _ltx2_get_sde_coeff 中的 NaN 处理逻辑。
4. 自动设置 text_encoder 后端：在 python/sglang/multimodal_gen/runtime/server_args.py 中，当 pipeline 为 LTX2 且后端非 DIFFUSERS 时，强制将

component_attention_backends['text_encoder'] 设为 torch_sdpa，并记录日志。

5. CI 工作流与 GT 管理：在 .github/workflows/diffusion-ci-gt-gen.yml 中扩展官方 GT 生成组，新增 ltx 组覆盖更多 case，更新 ci-data 引用支持指定分支，并调整 sparse checkout 以包含 repro 脚本；在 consistency_threshold.json 中调整 LTX-2.0 SSIM 阈值至 0.89。

关键文件：

- python/sglang/multimodal_gen/runtime/pipelines_core/stages/ltx_2_denoising.py（模块 去噪阶段；类别 source；类型 core-logic；符号 _ltx2_phi_scalar, _ltx2_get_res2s_coefficients_scalar, _ltx2_res2s_step_size_scalar）：核心 denoising stage，新增标量精度的 res2s 系数计算函数，是数值对齐的关键。
- python/sglang/multimodal_gen/runtime/models/adapter/ltx_2_connector.py（模块 适配器；类别 source；类型 data-contract；符号 _ltx2_connector_rope_freq_grid_np）：涉及 connector 中 RoPE 频率计算的精度提升，新增缓存函数。
- python/sglang/multimodal_gen/runtime/models/dits/ltx_2.py（模块 DiT 主干；类别 source；类型 data-contract；符号 _ltx2_rope_freq_grid_np）：DiT 主干中的 RoPE 频率计算对齐，与 connector 对称。
- python/sglang/multimodal_gen/runtime/models/encoders/gemma_3.py（模块 文本编码器；类别 source；类型 data-contract）：Gemma3 text encoder 注意力对齐，修改 mask 类型、GQA 处理和 RoPE 计算。
- python/sglang/multimodal_gen/runtime/server_args.py（模块 启动配置；类别 source；类型 core-logic）：自动设置 text_encoder 后端为 torch_sdpa，确保对齐生效。
- .github/workflows/diffusion-ci-gt-gen.yml（模块 CI 工作流；类别 infra；类型 infrastructure；符号 link）：CI 工作流扩展，增加官方 GT 生成覆盖和灵活性。
- python/sglang/multimodal_gen/runtime/pipelines/ltx_2_pipeline.py（模块 管道；类别 source；类型 core-logic）：管道级别调整，适配种子变化和精度对齐。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/denoising_av.py（模块 去噪阶段；类别 source；类型 core-logic）：音频 - 视频去噪通用调整，配合精度修改。
- python/sglang/multimodal_gen/test/test_utils.py（模块 测试工具；类别 test；类型 test-coverage）：测试工具更新，支持新的 GT 文件和 CI 场景。
- python/sglang/multimodal_gen/configs/sample/ltx_2.py（模块 配置样本；类别 source；类型 core-logic）：示例配置更新，增加官方 GT 相关配置。
- python/sglang/multimodal_gen/test/server/consistency_threshold.json（模块 一致性阈值；类别 test；类型 test-coverage）：调整 LTX-2.0 SSIM 阈值，平衡 CI 严格性。

关键符号：_ltx2_phi_scalar, _ltx2_get_res2s_coefficients_scalar, _ltx2_res2s_step_size_scalar, _ltx2_connector_rope_freq_grid_np, _ltx2_rope_freq_grid_np, Gemma3Attention.rotary_emb, Gemma3Attention.forward

关键源码片段

python/sglang/multimodal_gen/runtime/pipelines_core/stages/ltx_2_denoising.py

核心 denoising stage，新增标量精度的 res2s 系数计算函数，是数值对齐的关键。

```
# ltx_2_denoising.py 新增标量精度辅助函数

import math

@staticmethod
def _ltx2_phi_scalar(j: int, neg_h: float) -> float:
    # 计算标量版本的 phi 函数，避免张量运算中的舍入误差
    if abs(neg_h) < 1e-10:
        return 1.0 / math.factorial(j)
    remainder = sum(neg_h**k / math.factorial(k) for k in range(j))
    return (math.exp(neg_h) - remainder) / (neg_h**j)

@classmethod
def _ltx2_get_res2s_coefficients_scalar(
    cls, h: float, c2: float = 0.5
) -> tuple[float, float, float]:
    # 标量版本的 res2s 系数计算，与官方实现一致
    a21 = c2 * cls._ltx2_phi_scalar(1, -h * c2)
    b2 = cls._ltx2_phi_scalar(2, -h) / c2
    b1 = cls._ltx2_phi_scalar(1, -h) - b2
    return a21, b1, b2

@staticmethod
def _ltx2_res2s_step_size_scalar(
    sigma: torch.Tensor, sigma_next: torch.Tensor
) -> float:
    # 从张量中提取标量步长，保持高精度
    return float(
        (
            -torch.log(
                sigma_next.detach().double().cpu() / sigma.detach().double().cpu()
            )
        ).item()
    )
```

python/sglang/multimodal_gen/runtime/models/adapter/ltx_2_connector.py

涉及 connector 中 RoPE 频率计算的精度提升，新增缓存函数。

```
# ltx_2_connector.py 新增双精度 RoPE 频率计算
import functools
import numpy as np

@functools.lru_cache(maxsize=5)
def _ltx2_connector_rope_freq_grid_np(
    theta: float, num_pos_dims: int, dim: int
```

```
) -> torch.Tensor:
    # Official LTX uses NumPy float64 for double-precision RoPE frequencies.
    n_elem = 2 * num_pos_dims
    pow_indices = np.power(
        theta,
        np.linspace(0.0, 1.0, dim // n_elem, dtype=np.float64),
    )
    return torch.tensor(pow_indices * math.pi / 2.0, dtype=torch.float32)
```

评论区精华

gemini-code-assist[bot] 的 auto-review 指出关键风险：in-place tensor 修改可能破坏全局 sigma 调度或引起调用方副作用，并建议利用原生 SDPA 支持优化 GQA 而非手动扩展。PR 中已通过将 SDPA 范围限定在 text_encoder、采用 scalar 精度函数而非 in-place 修改等方式缓解了部分风险。

- In-place tensor operations risk (correctness): 开发者通过 scalar 函数和 NaN 处理避免 in-place，保留了原始路径
- SDPA GQA handling (performance): 开发者选择显式 repeat 以匹配官方行为，牺牲些许性能确保语义对齐

风险与影响

- 风险：
  1. Gemma3 注意力路径变更风险：在 gemma_3.py 中，attention mask 从 additive 改为 bool、GQA 从 enable_gqa 改为显式 repeat，虽然与官方对齐，但可能影响其他非 LTX 场景下的 Gemma3 编码器行为。由于该文件是 LTX-2 专有编码器，风险可控。
  2. RoPE 精度变更影响：_ltx2_rope_freq_grid_np 等函数改用 NumPy float64，在 double_precision 模式下可能引入微小数值差异，但已通过缓存和类型转换确保一致性。
  3. res2s 调度精度变更：新增的标量函数仅用于 HQ 等特定 case，且保留了原始张量路径，不影响主流。
  4. CI 工作流配置错误风险：diffusion-ci-gt-gen.yml 中新增的 ltx 组和 ci_data_ref 参数可能因权限或路径问题导致 GT 生成失败，但已有 fallback 机制（GET fallback for HEAD check）。 - 影响：用户影响：对使用 LTX-2.0/LTX-2.3 模型的最终用户，text encoder 输出和生成质量更接近官方，但无 breaking change；性能方面，torch_sdpa 仅用于 text_encoder，DiT 和 connector 仍可保持高性能后端。系统影响：CI 一致性测试更可靠，官方 GT 覆盖更多 case，阈值更严格，减少了误报。团队影响：为后续扩散模型精度对齐工作提供了可复用的模式（NumPy 双精度 RoPE 缓存、scalar 精度辅助函数）。
- 风险标记：核心路径变更，数值精度敏感，CI 配置复杂，多文件耦合

关联脉络

- PR #24320 [Misc] component attention backend override support: PR #24313 的 body 明确提到 merge of #24320，后者提供了组件级别 attention backend 覆盖机制，是

实现 text_encoder 单独使用 torch_sdpa 的基础。

- PR #23335 Fix diffusion fallback guards and validation: 同为 diffusion 模块的数值对齐相关工作，涉及 fallback 路径。
- PR #24117 [codex] Optimize Z-Image packed QKV: 同一模块（multimodal_gen）的性能优化 PR，与对齐工作互补。