PR #27443 完整报告

sgl-project/sglang

[Diffusion] Precompute Ideogram4 denoising metadata

合并时间：2026-06-07 14:27

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27443

执行摘要

- 一句话：预计算 Ideogram4 去噪元数据移除逐步开销
- 推荐动作：该 PR 是一个典型的“移动不变计算出循环”的优化案例，代码量小、逻辑清晰、profile 证据充分，适合作为性能优化参考。建议精读 _prepare_denoising_loop 和 _run_denoising_step 的对比，理解如何通过预计算消除逐步开销。

功能与动机

PR body 指出 profiler 跟踪显示 denoising 步内重复出现 aten::nonzero（690.480 ms / 8 calls）和 aten::cumsum 等掩码元数据构造操作，这些操作属于 host 可见的同步开销，可以通过预计算一次性消除。目标是将这些计算移至 _prepare_denoising_loop 阶段，使 denoising 循环主体更纯粹地执行 BF16 cast、GEMM、FlashAttention 和 RMSNorm 等 GPU 密集型计算。

实现拆解

1. 预计算 schedule 值和 delta（ideogram.py 的 _prepare_denoising_loop）：在去噪循环准备阶段调用 schedule(step_intervals) 获得 schedule_values，并计算 schedule_deltas = schedule_values[:-1] - schedule_values[1:]。将这两个张量存入 ctx.extra，替换原来存储的 schedule 函数引用和 step_intervals 张量。
2. 预构建 attention mask 和 varlen 元数据（同方法）：根据 data["segment_ids"] 和 neg_segment_ids 计算 attn_mask / neg_attn_mask（形状为 [batch_size, seq_len] 的布尔张量），并调用 build_varlen_mask_meta 得到 attn_mask_meta / neg_attn_mask_meta，一并放入 ctx.extra。
3. 简化 denoising 步内读取（_run_denoising_step）：从 ctx.extra 中直接取出 schedule_values 和 schedule_deltas，通过 t_val = schedule_values[i + 1]; t = t_val.expand(z.shape[0]) 替换原先的 float(schedule(...).item()) 重复求值。同时将预计算的 attn_mask 和 attn_mask_meta 传入 transformer forward。
4. Transformer forward 兼容性调整（ideogram.py 中 Ideogram4Transformer2DModel.forward）：签名增加两个可选参数 attn_mask: torch.Tensor | None = None 和 attn_mask_meta: dict | None = None。当它们为 None 时，仍以原有逻辑从 segment_ids 构造掩码和元数据，保持向后兼容。

关键文件：

- python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/ideogram.py（模块 扩散流程；类别 source；类型 core-logic；符号 _prepare_denoising_loop, _run_denoising_step）：Pipeline 核心逻辑：在 _prepare_denoising_loop 中预计算 schedule 值和 attention mask 元数据，并修改 _run_denoising_step 使用预计算值，是优化的主要载体。
- python/sglang/multimodal_gen/runtime/models/dits/ideogram.py（模块 扩散模型；类别 source；类型 data-contract；符号 Ideogram4Transformer2DModel.forward）：Transformer 模型定义：forward 签名增加可选参数以接受预计算的 mask 和元数据，同时保留 fallback 路径确保兼容性。

关键符号：Ideogram4DenoisingStage._prepare_denoising_loop, Ideogram4DenoisingStage._run_denoising_step, Ideogram4Transformer2DModel.forward

关键源码片段

python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/ideogram.py

Pipeline 核心逻辑：在 _prepare_denoising_loop 中预计算 schedule 值和 attention mask 元数据，并修改 _run_denoising_step 使用预计算值，是优化的主要载体。

```
# ideogram.py (pipeline stage)
# 在进入去噪循环前一次性预计算所有步间不变的元数据

def _prepare_denoising_loop(
    self, batch: Req, server_args: ServerArgs
) -> DenoisingContext:
    # ... 初始化 schedule、step_intervals、guidance_schedule ...

    # ★ 预计算 schedule 值和每步 delta，避免循环内反复求值
    schedule_values = schedule(step_intervals) # 张量，形状 [num_steps+1]
    schedule_deltas = schedule_values[:-1] - schedule_values[1:] # 相邻步差值

    # ... 构造 text_z_padding、neg_position_ids 等 ...

    # ★ 预计算 attention mask 及其 varlen 元数据（正 + 负）
    attn_mask = data["segment_ids"] > 0
    neg_attn_mask = neg_segment_ids > 0

    ctx.extra.update({
        "ideogram4_schedule_values": schedule_values, # 原先是函数引用 + intervals
        "ideogram4_schedule_deltas": schedule_deltas,
        "ideogram4_guidance_schedule": guidance_schedule,
        "ideogram4_text_z_padding": text_z_padding,
        "ideogram4_attn_mask": attn_mask,
        "ideogram4_attn_mask_meta": build_varlen_mask_meta(attn_mask), # 一次性构造
        # ... neg 版本同理 ...
```

```
        "ideogram4_neg_attn_mask": neg_attn_mask,
        "ideogram4_neg_attn_mask_meta": build_varlen_mask_meta(neg_attn_mask),
    })
    return ctx


def _run_denoising_step(
    self, batch: Req, step: DenoisingStepState, ctx: DenoisingContext
) -> DenoisingStepState:
    # ... 从 ctx.extra 中直接取出预计算值 ...
    schedule_values = ctx.extra["ideogram4_schedule_values"]
    schedule_deltas = ctx.extra["ideogram4_schedule_deltas"]
    guidance_schedule = ctx.extra["ideogram4_guidance_schedule"]
    # ...
    i = step.t_int
    t_val = schedule_values[i + 1] # 直接索引，取代 schedule(step_intervals).item()
    t = t_val.expand(z.shape[0]) # expand 替代逐元素 full
    # ... 调用 transformer forward 时传入预构建的 mask 元数据 ...
    pos_z = torch.cat([ctx.extra["ideogram4_text_z_padding"], z], dim=1)
    # ...
    model_output = self.model[
        "unet"
    ].forward(
        llm_features=neg_llm_features,
        x=neg_z,
        t=t,
        position_ids=data["position_ids"],
        segment_ids=data["segment_ids"],
        indicator=data["indicator"],
        attn_mask=ctx.extra["ideogram4_attn_mask"], # 传递预计算掩码
        attn_mask_meta=ctx.extra["ideogram4_attn_mask_meta"],
    )
    # ...
```

python/sglang/multimodal_gen/runtime/models/dits/ideogram.py

Transformer 模型定义：forward 签名增加可选参数以接受预计算的 mask 和元数据，同时保留 fallback 路径确保兼容性。

```
# ideogram.py (transformer model)
# forward 签名增加可选参数


def forward(
    self,
    *,
    llm_features: torch.Tensor,
    x: torch.Tensor,
    t: torch.Tensor,
    position_ids: torch.Tensor,
    segment_ids: torch.Tensor,
```

```
    indicator: torch.Tensor,
    attn_mask: torch.Tensor | None = None, # ★ 新增可选参数
    attn_mask_meta: dict | None = None, # ★ 新增可选参数
    **kwargs,
) -> torch.Tensor:
    # ... 类型转换和投影 ...
    cos, sin = self.rotary_emb(h, position_ids)

    # 构建注意力掩码：如果外部传入了预计算版本则直接使用，否则内部 fallback
    if attn_mask is None:
        attn_mask = segment_ids > 0
    if attn_mask_meta is None:
        attn_mask_meta = build_varlen_mask_meta(attn_mask) # 仅在需要时构造

    for layer in self.layers:
        h = layer(
            h,
            cos=cos,
            sin=sin,
            adaln_input=adaln_input,
            attn_mask=attn_mask,
            attn_mask_meta=attn_mask_meta,
        )
    return self.final_layer(h, c=adaln_input).to(torch.float32)
```

评论区精华

本 PR 无实质 review 讨论。仅有一位自动化 reviewer（gemini-code-assist）给出"无 review 评论"的确认，以及维护者 mickqian 的两次 approve。

- 暂无高价值评论线程

风险与影响

- 风险：风险低。变更集中在对性能无负面影响的数据预计算和接口扩展：
  - 回归风险：如果未来有新增的 denoising 步内逻辑依赖 context 中已移除的旧 key（如 ideogram4_schedule 函数或 ideogram4_step_intervals），可能引发 key error。但 PR 已同时修改了消费者 _run_denoising_step，且同文件内均为内部调用，无外部依赖。
  - 兼容性风险：Ideogram4Transformer2DModel.forward 新参数设为可选，旧调用方不传参时行为完全不变，无破坏性变化。
  - 性能风险：预计算耗时仅在去噪循环前执行一次，对整体延迟贡献极小。
  - 影响：影响范围：仅限 Ideogram4 去噪流程，不涉及其他 pipeline 或模型。影响程度：较小。profile 确认消除了 host-visible 掩码构造热点，但端到端收益仅约 0.12%，属于微观优化。对于高分辨率或长步数场景，收益可能略有放大。用户可见性：不影响生成质量或输出格式；用户无需修改任何配置。
- 风险标记：预计算值 key 变更需确保消费者同步更新

关联脉络

- 暂无明显关联 PR