PR #27041 完整报告

sgl-project/sglang

[diffusion] Optimize Cosmos3 lossless hot paths

合并时间：2026-06-02 19:47

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27041

执行摘要

- 一句话：优化 Cosmos3 去噪热点路径，集中进度条、融合 RoPE 与 QK 归一化
- 推荐动作：建议仔细阅读 qwen3_apply_rotary_pos_emb 的重写，学习如何在保持数值等价下通过内联和直接写入减少张量操作；进度条集中化模式适用于其他子模块；_postprocess_tensor 的 in-place 技巧可推广到类似场景。该 PR 展示了典型的“测量 - 识别热点 - 优化”流程，值得工程团队参考。

功能与动机

根据 PR 描述，本次变更旨在优化 Cosmos3 视频生成的无损热点路径，降低响应延迟和峰值内存，同时保持输出精度。具体优化点包括进度条集中化、RoPE 计算内联、QK 归一化融合和 VAE 解码与后处理减少临时分配。

实现拆解

1. 进度条集中化：在 python/sglang/multimodal_gen/runtime/pipelines_core/stages/base.py 的 PipelineStage 基类添加 progress_bar 方法，通过 get_world_rank() != 0 控制仅在全局 rank 0 显示。从 denoising.py 和 mova.py 移除各自的本地 progress_bar 实现，并清理相关导入（tqdm、Iterable 等）。在 cosmos3.py 的去噪循环中改为调用 self.progress_bar。
2. RoPE 计算内联：在 python/sglang/multimodal_gen/runtime/models/dits/cosmos3video.py 中重写 qwen3_apply_rotary_pos_emb，不再调用 qwen3_rotate_half（该函数被删除），而是将 q/k 分别拆为两半，直接通过逐元素运算写入预分配的空张量，避免 torch.cat 带来的额外 kernel launch 和内存复制。
3. 融合 QK 归一化：在 cosmos3video.py 中导入 apply_qk_norm（来自 sglang.multimodal_gen.runtime.layers.layernorm），将交叉注意力入口处两行独立的 F.rms_norm 替换为单次 apply_qk_norm 调用，减少函数调用开销和 kernel launch 次数。
4. WanVAE 解码优化：在 python/sglang/multimodal_gen/runtime/models/vaes/wanvae.py 的 decode 方法中，使用 out_chunks 列表替代原有的 outs 列表，并移除冗余的 first_chunk 条件分支（简化为 first_chunk.set(i==0)），最终 torch.cat 前检查列表长度，避免单元素时不必要的 concat。
5. 后处理 In-place 操作：在 python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/cosmos3.py 的 _postprocess_tensor 中使用 mul_、add_、

clamp_ 原地修改张量，减少临时内存分配和释放。

关键文件：

- python/sglang/multimodal_gen/runtime/models/dits/cosmos3video.py（模块 扩散模型；类别 source；类型 core-logic；符号 qwen3_rotate_half, qwen3_apply_rotary_pos_emb, apply_qk_norm）：核心热点重写：RoPE 内联优化（删除 qwen3_rotate_half、重写 qwen3_apply_rotary_pos_emb）以及使用 fused apply_qk_norm，直接影响每秒数十次的交叉注意力前向计算。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/base.py（模块 流水线基础；类别 source；类型 core-logic；符号 progress_bar）：新增 progress_bar 方法统一管理进度条，控制仅在全局 rank 0 显示，消除各阶段重复实现。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/denoising.py（模块 去噪阶段；类别 source；类型 core-logic；符号 progress_bar）：移除本地的 progress_bar 实现，改用基类方法，并清理相关导入。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/mova.py（模块 MOVA 阶段；类别 source；类型 data-contract；符号 progress_bar）：移除本地的 progress_bar 实现，改用基类方法，并清理相关导入。
- python/sglang/multimodal_gen/runtime/models/vaes/wanvae.py（模块 VAE 模型；类别 source；类型 data-contract；符号 decode）：优化 VAE 解码循环：简化条件分支，使用列表缓存代替重复 cat，减少内存开销。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/cosmos3.py（模块 Cosmos3 阶段；类别 source；类型 data-contract；符号 _postprocess_tensor）：后处理函数改为 in-place 操作，减少临时内存；去噪循环改用基类 progress_bar。

关键符号：progress_bar, qwen3_apply_rotary_pos_emb, _postprocess_tensor, decode

关键源码片段

python/sglang/multimodal_gen/runtime/models/dits/cosmos3video.py

核心热点重写：RoPE 内联优化（删除 qwen3_rotate_half、重写 qwen3_apply_rotary_pos_emb）以及使用 fused apply_qk_norm，直接影响每秒数十次的交叉注意力前向计算。

```
defqwen3_apply_rotary_pos_emb(     q: torch.Tensor,     k: torch.Tensor,     cos: torch.Tensor,     sin: torch.Tensor, ) -> tuple[torch.Tensor, torch.Tensor]: " ""Qwen3-style RoPE ： 避免 concat，直接写入两半。 将 q / k 拆为前后两半，分别计算并写入预分配空张量。 """     half = q.shape[-1] // 2     q1 = q[..., :half]     q2 = q[..., half:]     q_embed = torch.empty_like(q)     # 前半 ：  q1 * cos - q2 * sin     q_embed[..., :half] = q1 * cos[..., :half] - q2 * sin[..., :half]     # 后半 ：  q2 * cos + q1 * sin     q_embed[..., half:] = q2 * cos[..., half:] + q1 * sin[..., half:]     half = k.shape[-1] // 2     k1 = k[..., :half]     k2 = k[..., half:]     k_embed = torch.empty_like(k)     k_embed[..., :half] = k1 * cos[..., :half] - k2 * sin[..., :half]     k_embed[..., half:] = k2 * cos[..., half:] + k1 * sin[..., half:]     return q_embed, k_embed （注：函数体完全重写，原辅助函数
```

qwen3_rotate_half 被删除）

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

新增 progress_bar 方法统一管理进度条，控制仅在全局 rank 0 显示，消除各阶段重复实现。

defprogress_bar(self, iterable: Iterable | None = None, total: int | None = None, *, disable: bool = False, **kwargs,) -> tqdm: """创建进度条，仅在全局 rank 0 显示。""" return tqdm(iterable=iterable, total=total, disable=disable or get_world_rank() != 0, **kwargs,) （新增方法，供各阶段子类调用）

评论区精华

唯一的 review 来自 gemini-code-assist[bot]，在 cosmos3video.py 的 qwen3_apply_rotary_pos_emb 建议：由于 cos 和 sin 的前后半相同，可以只切片一次 cos[..., :half] 和 sin[..., :half] 并在 q 和 k 的计算中重用，从而避免 6 次冗余 slice 操作。作者尝试了 commit "Reuse Cosmos3 RoPE trig slices" 但随后 revert，可能因为可读性或数值精度感知的考虑。该建议最终未被采纳。

- 进一步优化 RoPE 切片以避免冗余 slicing (performance): 未采纳，相关 commit 被 revert ，可能由于可读性或数值精度考量。

风险与影响

- 风险：变更集中在热点路径，但验证输出字节一致（sha256 相同），数值回归风险极低。in-place 后处理 (mul_ 等）在推理场景无梯度影响，但若未来需要反传需注意。进度条分布式控制从 local_rank 改为 get_world_rank()，在标准分布式设置中行为一致，但极端非标准配置（如主进程 rank 非 0）可能表现意外。整体风险可控。
- 影响：对用户：Cosmos3 视频生成任务响应时间降低约 1.6%，峰值内存减少 3.9%，输出不变，体验提升。对系统：多卡分布式环境下进度条仅在 rank 0 打印，减少日志冗余。对团队：无直接开发影响，但引入 apply_qk_norm 依赖需确保其他模型（如 LingBot）在更新后仍正常工作。
- 风险标记：热点路径变更，无新增测试（仅验证输出一致），变更分散多文件

关联脉络

- PR #27037 [diffusion] Enable Cosmos3 parallel decode: 同一功能线（Cosmos3 性能优化），修改了 cosmos3.py 和 cosmos3video.py 等文件，构成了 Cosmos3 持续性能改进的一部分。
- PR #26973 [diffusion] reduce Cosmos3 denoise overhead: 前一版本性能优化，同样涉及 cosmos3.py 和 cosmos3video.py，本 PR 在其基础上进一步优化热点路径。