PR #34932 完整报告

sgl-project/sglang

[diffusion] Accelerate Cosmos3 T2I QKNorm+RoPE

合并时间：2026-08-16 20:15

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34932

执行摘要

- 一句话：Cosmos3 T2I 接入 fused QKNorm+RoPE，B300 提速约 13.4%
- 推荐动作：值得精读：它展示了如何在性能优化中保住 bit-exact 语义，并用显式守卫（CUDA / 非编译 / 非 SP / _gen_layers_torch_compiled）控制路径切换。关注 can_use_fused_inplace_qknorm_rope 的 gate 条件以及缓存 dtype 提前收敛的细节，这些是同类优化可复用的模式。

功能与动机

PR body 指出 Cosmos3 video 已经使用 fused QKNorm + RoPE，但 T2I 因 image 路径需要中间 BF16 round 而停留在独立 QKNorm、RoPE、KV 拼接操作上。共享 fused kernel 现已支持该 rounding 模式，因此本 PR 将 Cosmos3 T2I 接入 fused 路径，并避免逐层 cache cast 与多次 KV packing launch。

实现拆解

实现分三步：

1. 给 fused 路径透传取整开关：在 python/sglang/multimodal_gen/runtime/models/dits/cosmos3video.py 中，为 _apply_qwen3_qk_norm_rope、_apply_qwen3_qk_norm_rope_pack_kv 以及 GEN 注意力层 forward 新增关键字参数 round_norm_before_rope（默认 False），并透传给 apply_qk_norm_rope / fused_qknorm_rope_pack_kv，使 fused 内核能复现 split 路径的 BF16 取整语义。
2. 在模型主干启用 T2I fused 路径：Cosmos3 主干 forward 中，round_norm_before_rope = T == 1；use_fused_qk_norm_rope 在 T > 1 或满足 CUDA、非 torch.compile、未对 gen_layers 编译、单 SP、can_use_fused_inplace_qknorm_rope 校验时开启。同时在 T == 1 且未编译时，将 build_rope_cache_inputs 返回的 rounded cos_sin_gen 提前 .to(hidden_gen.dtype) 存入 cached_gen_rope_inputs，避免每个 GEN layer 重复 cast。
3. 标记编译状态并补回归测试：在 python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/cosmos3.py 的 _maybe_enable_torch_compile 中设置 transformer._gen_layers_torch_compiled = True，使编译后的 GEN layers 继续走 compile-safe 的 split 路径；在 test/registered/kernels/ops/diffusion/test_qknorm_rope.py 新增 test_qknorm_rope_pack_kv_preserves_split_bf16_rounding，用 torch.equal 验

证 fused 路径与 split 路径逐位一致。

关键文件：

- python/sglang/multimodal_gen/runtime/models/dits/cosmos3video.py（模块 模型层；类别 source；类型 core-logic；符号 _apply_qwen3_qk_norm_rope, _apply_qwen3_qk_norm_rope_pack_kv, forward）：核心模型改动：新增 round_norm_before_rope 参数并透传 fused kernel，为主干 forward 启用 T2I fused QKNorm+RoPE+KV-pack 路径，同时提前收敛 RoPE 缓存 dtype。
- test/registered/kernels/ops/diffusion/test_qknorm_rope.py（模块 内核测试；类别 test；类型 test-coverage；符号 test_qknorm_rope_pack_kv_preserves_split_bf16_rounding）：新增回归测试，用 split 路径（fused_inplace_qknorm + rotary_embedding）作为参考，验证 fused_qknorm_rope_pack_kv 在 round_norm_before_rope=True 时输出逐位相等，是 bit-exact 契约的关键保障。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/cosmos3.py（模块 流水线；类别 source；类型 core-logic；符号 _maybe_enable_torch_compile）：在 torch.compile GEN layers 时设置 transformer._gen_layers_torch_compiled 标志，确保编译场景继续走 compile-safe 的 split 路径，避免 fused inplace kernel 进入编译图。

关键符号：_apply_qwen3_qk_norm_rope, _apply_qwen3_qk_norm_rope_pack_kv, forward, test_qknorm_rope_pack_kv_preserves_split_bf16_rounding

关键源码片段

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

核心模型改动：新增 round_norm_before_rope 参数并透传 fused kernel，为主干 forward 启用 T2I fused QKNorm+RoPE+KV-pack 路径，同时提前收敛 RoPE 缓存 dtype。

```
# Cosmos3 主干 forward 中，针对 T == 1（T2I）启用共享 fused 路径的关键分支。
# round_norm_before_rope 保证 fused 内核先做 BF16 取整再做 RoPE，
# 从而与原有 split 路径逐位一致（bit-exact）。
round_norm_before_rope = T == 1
use_fused_qk_norm_rope = T > 1 or (
    hidden_gen.device.type == "cuda"
    and not torch.compiler.is_compiling()
    and not self._gen_layers_torch_compiled
    and get_sp_world_size() == 1
    and can_use_fused_inplace_qknorm_rope(
        self.head_dim,
        cos_sin_gen.shape[-1],
        True,
        hidden_gen.dtype,
        cos_sin_gen.dtype,
        round_norm_before_rope=True,
        pack_kv=True,
    )
)
```

```
# T == 1 且未编译时，cos_sin_cache 已按激活 dtype 完成取整，
# 提前落为激活 dtype，避免每个 GEN layer 重复 cast。
if T == 1 and not self._gen_layers_torch_compiled:
    cos_sin_gen = cos_sin_gen.to(hidden_gen.dtype)
self.cached_gen_rope_inputs[cache_key] = (cos_sin_gen, gen_rope_cache_positions)

for i, layer in enumerate(self.gen_layers):
    k_und, v_und = cached_kv_for_key[i]
    hidden_gen, residual = layer(
        hidden_gen,
        k_und,
        v_und,
        cos_sin_gen,
        gen_rope_cache_positions,
        use_fused_qk_norm_rope,
        round_norm_before_rope,
        residual=residual,
    )
```

test/registered/kernels/ops/diffusion/test_qknorm_rope.py

新增回归测试，用 split 路径（fused_inplace_qknorm + rotary_embedding）作为参考，验证 fused_qknorm_rope_pack_kv 在 round_norm_before_rope=True 时输出逐位相等，是 bit-exact 契约的关键保障。

```
# 新增回归测试：验证 fused_qknorm_rope_pack_kv 在 round_norm_before_rope=True 时，
# 与“split 路径”（fused_inplace_qknorm + rotary_embedding）逐位一致。
def test_qknorm_rope_pack_kv_preserves_split_bf16_rounding() -> None:
    from sgl_kernel import rotary_embedding
    from sglang.kernels.ops.diffusion.qknorm_rope import fused_qknorm_rope_pack_kv
    from sglang.kernels.ops.layernorm.norm import fused_inplace_qknorm

    # 构造 suffix 的 qkv 与 prefix 的 k/v，模拟 GEN 层到缓存 UND K/V 的 cross-attention
    qkv = torch.randn(batch_size, suffix_tokens, num_heads, head_dim, device=DEVICE, dtype=
    DTYPE)
    k_prefix = prefix_qkv[:, :, num_q_heads : num_q_heads + num_kv_heads]
    v_prefix = prefix_qkv[:, :, num_q_heads + num_kv_heads :]

    # 参考实现：先 QKNorm，再 rotary_embedding，最后 cat 上 prefix，得到 packed_k_ref /
    packed_v_ref
    fused_inplace_qknorm(
        q_ref.view(-1, num_q_heads, head_dim),
        k_ref.view(-1, num_kv_heads, head_dim),
        q_weight,
        k_weight,
        eps=1e-6,
    )
    rotary_embedding(positions, q_ref.view(-1, num_q_heads * head_dim), k_ref.view(-1, num_kv_
    heads * head_dim), head_dim, cos_sin_cache, True)
    packed_k_ref = torch.cat([k_prefix, k_ref], dim=1)
```

```
    packed_v_ref = torch.cat([v_prefix, v_ref], dim=1)

    # 被测试实现：一次 kernel 调用完成 QKNorm + RoPE + KV pack，并开启 BF16 取整
    fused_qknorm_rope_pack_kv(
        q_fused, k_fused, v_fused, k_prefix, v_prefix, packed_kv,
        q_weight, k_weight, cos_sin_cache, positions,
        is_neox=True, rope_dim=head_dim, round_norm_before_rope=True,
    )

    # 逐位相等断言：fused 路径必须完整复现 split 路径的取整语义
    assert torch.equal(q_ref, q_fused)
    assert torch.equal(packed_k_ref, packed_kv[0])
    assert torch.equal(packed_v_ref, packed_kv[1])
```

python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/cosmos3.py

在 torch.compile GEN layers 时设置 transformer._gen_layers_torch_compiled 标志，确保编译场景继续走 compile-safe 的 split 路径，避免 fused inplace kernel 进入编译图。

```
# torch.compile 场景下，为 transformer 打上编译标记，
# 模型 forward 据此避开 fused inplace kernel，继续走 compile-safe 的 split 路径。
if gen_layers is not None and isinstance(gen_layers, nn.ModuleList):
    logger.info("Compiling %d Cosmos3 gen_layers with %s", len(gen_layers), compile_kwargs)
    transformer._gen_layers_torch_compiled = True
    for i, layer in enumerate(gen_layers):
        gen_layers[i] = torch.compile(layer, **compile_kwargs)
else:
    logger.warning("Cosmos3 gen_layers not found, skipping torch.compile")
```

评论区精华

该 PR 无 review 评论，作者直接合并，公开讨论线程为空。设计权衡主要体现在代码本身：round_norm_before_rope 默认 False 保证 video 路径行为不变；_gen_layers_torch_compiled 显式区分 torch.compile 场景，避免 fused inplace kernel 在编译图内执行。

- 暂无高价值评论线程

风险与影响

- 风险：主要风险集中在 fused 路径的 bit-exact 语义和平台 / 编译守卫上：
  - cosmos3video.py 中 T2I 默认路径从 split 切换到 fused，若 fused kernel 的 round_norm_before_rope 实现与 *eager* 语义有细微差异，会引起输出漂移；新增测试仅覆盖单一 dtype（测试常量 DTYPE），其他 dtype 未验证。
  - _gen_layers_torch_compiled 是运行时实例属性，若用户绕过 stage 入口直接对 gen_layers 手动 torch.compile，该标志不置位，可能误入 fused 路径导致编译错误。

- 非 CUDA 平台、多 SP、torch.compiler.is_compiling() 场景会自动回退到 split 路径，正确性风险低，但这些场景拿不到性能收益。
- 影响：影响面集中在 Cosmos3 T2I（T == 1）lossless *eager* 用户，B300 上 *denoise* 从约 0.997 s 降到 0.863 s，E2E 约 0.908 s；使用 torch.compile 或 video（T > 1）的路径行为不变。该模式为后续 diffusion 模型复用 fused QKNorm + RoPE 的 rounding 语义提供了可参考的开关设计，同时 kernel 层回归测试保护了 bit-exact 契约。
- 风险标记：核心路径变更，T2I 默认路径切换，Bit-exact 依赖 fused kernel，手动 torch.compile 场景标志可能缺失，非 CUDA / 多 SP 无收益

关联脉络

- PR #34931 [diffusion] Accelerate lossless Ideogram norm post-processing: 同属 diffusion 性能优化系列，都通过复用共享 kernel 减少重复 launch 来提速 lossless 路径。
- PR #34929 [diffusion] Enable breakable CUDA graphs for LTX-2.3: 同属 diffusion 加速改动，涉及 torch.compile / CUDA graph 路径选择，与 Cosmos3 编译守卫场景互补。
- PR #34949 [Diffusion] Route MiniMax H3 VAE attention through native backends: 同属 diffusion 注意力后端统一工作，将注意力路由到原生后端，为本 PR 的 fused 路径复用共享 kernel 提供基础。