PR #34314 完整报告

sgl-project/sglang

[diffusion] Ideogram-4: fuse Qwen3-style RoPE and SwiGLU silu-mul (denoise -5.1% H100 / -4.7% H200, bit-exact)

合并时间：2026-08-12 09:19

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34314

执行摘要

- 一句话：Ideogram-4 融合 RoPE 与 SwiGLU，denoise 降约 5%
- 推荐动作：值得精读。这是 diffusion 模块性能优化链条（#34305 → #34306 → 本 PR）的收尾一环，价值在于：① 单 kernel 融合如何逐边界复刻 *eager* 的 bf16 舍入（round_bf16_to_fp32 + store 时二次舍入），是写数值敏感融合 kernel 的范本；② BitExactFusionGate 首调自校验 + 永久回退的挂载模式，让 bit-exact 优化可以默认开启而零风险，值得在更多模型推广；③ 放弃拼接输入 silu_and_mul（避免全宽 cat 抵消收益）的取舍分析很有参考价值。建议结合 #34305、#34306 与 ERNIE 的挂载方式一起阅读。

功能与动机

PR body 指出：在 #34305 移除 per-forward FP8 权重反量化后，Ideogram-4 在 H100 上 *eager* 与 *compile* 的剩余差距来自其自身的 *elementwise* 运算链（TP=2 trace 中约占每步 GPU 时间的 15%）。Qwen3 风格 rotate-half RoPE 每层每个投影要跑约 6 个 kernel（四次乘法、两次加 / 减，外加 qwen3_apply_rotary_pos_emb 中切片 *empty_like* 的填充）；SwiGLU 的 F.silu(w1(x)) * w3(x) 对 FFN 中间结果做两遍全量扫描，而 gate/up 投影是分开的 GEMM，拼接输入的 silu_and_mul kernel 需要额外一次全宽 cat 传递，收益被抵消。两条链都是纯 *elementwise*，因此可以逐边界复刻 aten 的 bf16 舍入，在不改数值结果的前提下融合提速。

实现拆解

1. 新增双输入 silu(a) * b Triton 融合 kernel（python/sglang/kernels/ops/diffusion/triton/silu_mul_bitexact.py，新增 78 行）：_silu_mul_kernel 单 kernel 完成加载、a * tl.sigmoid(a)、舍入与 * b；通过 round_bf16_to_fp32 复刻 *eager* F.silu 输出的 bf16 舍入（第一次舍入），store 到 bf16 输出时自然完成乘法结果的第二次（也是最后一次）舍入，与 *eager* 两 kernel 链的舍入边界一一对应。can_use_fused_silu_mul 前置校验 CUDA、bf16、同设备同形状、连续、非空；fused_silu_mul_bitexact 经 @register_custom_op 注册，并配套 _fake_silu_mul（torch.empty_like）fake 实现，保证 torch.compile 阶段的元数据推断可用。选择双输入 kernel 而非拼接输入 silu_and_mul，是因为 gate/up 是分开的 GEMM，拼接需要多一次全宽 cat，收益会被抵消。
2. 在 Ideogram-4 主路径挂载融合路径（python/sglang/multimodal_gen/runtime/models/dits/ideogram.py，+118/-2）：新增 _can_use_fused_rope 校验（bf16、CUDA、4D、连

续、cos/sin 为 (B, S, 1, D) full-span 行且连续、head_dim 为偶数）与 _ideogram_rope，将 cos/sin reshape 为行向量后对 q、k 各调一次 fused_rope_rotate_half_bitexact，其位级等价性依赖 round-to-nearest 下 round(x) + round(-y) 与 round(x) - round(y) 严格相等；新增 _ideogram_swiglu 走 fused_silu_mul_bitexact。两个入口都挂在模块级 BitExactFusionGate（_IDEOGRAM_ROPE / _IDEOGRAM_SWIGLU）之后，首次调用把 fused 结果与 *eager reference* 用 torch.equal 在真实张量上对比，一致则永久放行，不一致或抛异常则永久回退 *eager*。注入点替换：Ideogram4Attention.forward 中 qwen3_apply_rotary_pos_emb(q, k, cos, sin) → _ideogram_rope(q, k, cos, sin)；Ideogram4MLP.forward 中 F.silu(self.w1(x)) * self.w3(x) → _ideogram_swiglu(self.w1(x), self.w3(x))。

3. 测试与验证配套（python/sglang/multimodal_gen/test/unit/test_ideogram_rope_swiglu_fusion.py，新增 63 行）：CUDA 路径用 torch.equal 对比 fused 与 *eager*，RoPE 覆盖 (2, 257, 16, 128) 与 (1, 64, 3, 64) 两组形状，SwiGLU 覆盖 (2, 513, 3584) bf16 输入；CPU 路径验证 *gate* 禁用 / 条件不满足时直接返回 *eager* 结果。性能验证：ideogram4-fp8 preset、TP=2、1024²、--quality=lossless 下，H200 *denoise* 5.005s → 4.769s（-4.7%）、端到端 5.211s → 4.898s，H100 *denoise* 5.177s → 4.911s（-5.1%）、端到端 5.315s → 5.041s，两张卡输出 md5 与 main 完全一致，*gate* 日志零 fallback。

关键文件：

- python/sglang/multimodal_gen/runtime/models/dits/ideogram.py（模块 图片生成；类别 source；类型 core-logic；符号 _can_use_fused_rope, _ideogram_rope, _ideogram_swiglu）：主路径挂载点：在 Ideogram4Attention.forward 与 Ideogram4MLP.forward 中用 _ideogram_rope / _ideogram_swiglu 替换 *eager* 链，并引入 BitExactFusionGate 自校验与永久回退，是本 PR 行为契约的核心。
- python/sglang/kernels/ops/diffusion/triton/silu_mul_bitexact.py（模块 融合算子；类别 source；类型 jit-kernel；符号 _silu_mul_kernel, can_use_fused_silu_mul, _fake_silu_mul, fused_silu_mul_bitexact）：新增双输入 silu(a) * b Triton 融合 kernel，复刻 *eager* 两 kernel 链的 bf16 舍入边界，并注册 custom op 与 fake 实现；与 #34306 的 rope_rotate_half_bitexact.py 一起构成可复用的 bit-exact 融合工具。
- python/sglang/multimodal_gen/test/unit/test_ideogram_rope_swiglu_fusion.py（模块 单元测试；类别 test；类型 test-coverage；符号 _make_qk_cos_sin, TestIdeogramRopeFusion, test_fused_rope_matches_eager, test_eager_fallback_cpu）：新增 fused 与 *eager* 的 torch.equal 逐位对比测试以及 CPU fallback 测试，是 bit-exact 保证的直接回归防线。

关键符号：_can_use_fused_rope, _ideogram_rope, _ideogram_swiglu, fused_silu_mul_bitexact, can_use_fused_silu_mul, _silu_mul_kernel, _fake_silu_mul

关键源码片段

python/sglang/kernels/ops/diffusion/triton/silu_mul_bitexact.py

新增双输入 silu(a) * b Triton 融合 kernel，复刻 *eager* 两 kernel 链的 bf16 舍入边界，并注册 custom op 与 fake 实现；与 #34306 的 rope_rotate_half_bitexact.py 一起构成可复用的

bit-exact 融合工具。

```
# SPDX-License-Identifier: Apache-2.0
# 双输入 silu(a) * b 的 bit-exact 融合 kernel。
# SwiGLU MLP 的 gate/up 投影是各自独立的 GEMM，拼接输入的 silu_and_mul
# 需要额外一次全宽 cat 传递而收益被抵消，所以这里用双输入 kernel 把
# eager 的 F.silu(a)（一次 kernel）与 s * b（又一次 kernel）合并为一遍，
# 并逐边界复刻 aten 的 bf16 舍入：
# - silu 是单次 aten 算子（fp32 运算、一次舍入），tl.sigmoid 与 aten
# 使用同一个 fp32 sigmoid，确认在 100 万随机 bf16 值上逐位一致；
# - 乘法在 store 到 bf16 输出时再舍入一次，与 eager 的 s * b 相同。
# 调用方仍会做首次调用自校验，不一致即回退 eager。

from __future__ import annotations

import torch
import triton # type: ignore
import triton.language as tl # type: ignore

from sglang.kernels.ops.diffusion.triton.numerics import round_bf16_to_fp32
from sglang.srt.utils.custom_op import register_custom_op


@triton.jit
def _silu_mul_kernel(out_ptr, a_ptr, b_ptr, numel, BLOCK: tl.constexpr):
    # 每个 program 处理 BLOCK 个元素，越界位置以 0 填充且不会写回。
    offs = tl.program_id(0).to(tl.int64) * BLOCK + tl.arange(0, BLOCK)
    mask = offs < numel
    # 输入是 bf16，转 fp32 后做 fp32 运算，保证与 aten 的运算精度一致。
    a = tl.load(a_ptr + offs, mask=mask, other=0.0).to(tl.float32)
    b = tl.load(b_ptr + offs, mask=mask, other=0.0).to(tl.float32)
    # 第一次舍入：silu 结果按 bf16 舍入后再回 fp32，复刻 eager F.silu 的输出 dtyple。
    s = round_bf16_to_fp32(a * tl.sigmoid(a))
    # 第二次舍入：乘积直接 store 到 bf16 输出，等价于 eager 中 s * b 的舍入。
    tl.store(out_ptr + offs, s * b, mask=mask)


def can_use_fused_silu_mul(a: torch.Tensor, b: torch.Tensor) -> bool:
    # 仅 CUDA、bf16、同设备同形状、连续非空的输入走融合路径。
    return (
        a.dtype is torch.bfloat16
        and b.dtype is torch.bfloat16
        and a.is_cuda
        and b.is_cuda
        and a.device == b.device
        and a.shape == b.shape
        and a.is_contiguous()
        and b.is_contiguous()
        and a.numel() > 0
```

```
    )


def _fake_silu_mul(a: torch.Tensor, b: torch.Tensor) -> torch.Tensor:
    # torch.compile 阶段只关心 shape/dtype 元数据，无需真实计算。
    return torch.empty_like(a)


@register_custom_op(
    op_name="triton_fused_silu_mul_bitexact",
    mutates_args=[],
    fake_impl=_fake_silu_mul,
)
def fused_silu_mul_bitexact(a: torch.Tensor, b: torch.Tensor) -> torch.Tensor:
    # 单次 grid 启动，BLOCK 固定为 1024，一次覆盖所有元素。
    out = torch.empty_like(a)
    numel = a.numel()
    with torch.cuda.device(a.device):
        _silu_mul_kernel[(triton.cdiv(numel, 1024),)](
            out,
            a,
            b,
            numel,
            BLOCK=1024,
        )
    return out
```

评论区精华

该 PR 全程无 review 评论（0 条 issue 评论、0 条 review 评论），合并前无人工讨论记录，设计论证集中在 PR body，核心决策有三点：① 为什么不用拼接输入的 silu_and_mul——Ideogram-4 的 gate/up 是分开的 GEMM，拼接需要额外一次全宽 cat 传递，收益被抵消，因此选择双输入 silu(a) * b kernel；② 位级一致的数学依据——round(round(q1*cos1) + round(-q2*sin1)) 与 eager 的 round(round(q1*cos1) - round(q2*sin1)) 在 round-to-nearest 下严格等价，tl.sigmoid 与 aten silu 在 100 万随机 bf16 值上逐位一致；③ 默认挂载而非 A/B 开关——通过 BitExactFusionGate 首调自校验 + 永久 eager 回退保证安全，实现了「默认开启、零精度风险」。

- 暂无高价值评论线程

风险与影响

- 风险：
  - 位级一致性依赖数值实现：tl.sigmoid 与 aten silu 的逐位一致、round_bf16_to_fp32 的舍入语义以及 Triton 编译器对 fp32 中间精度的假设，任何一项在 Triton/aten 升级后变化都可能破坏 bit-exact 前提。BitExactFusionGate 会在运行期捕获并永久回退，但首次调用有一次对比开销，回退后性能收益消失，建议发版前跑 md5 级数值回归。

- 默认挂载的行为契约变更：ideogram.py 替换的是 Ideogram4Attention.forward 与 Ideogram4MLP.forward 主路径，新增形状 / 硬件组合（如 Blackwell、AMD、NPU）未被验证时也会默认尝试融合，依赖 gate 现场发现；性能数据只覆盖 H100/H200。
- 测试覆盖边界：CUDA 测试依赖 torch.cuda.is_available()，纯 CPU CI 只覆盖 fallback 分支；RoPE 只测了 head_dim=128/64 的偶数场景，_can_use_fused_rope 中非连续、形状不匹配等拒绝分支无直接单测。
- 跨 PR 共享文件：rope_rotate_half_bitexact.py 与 #34306（ERNIE）共用且内容一致、可任意顺序合并，后续修改会同时影响两个模型，需要留意同步。
- 首次调用自校验会引入一次 eager 对比的临时开销，在极短会话 / 小 batch 场景可能吃掉部分收益（生产运行中 zero fallback，实际可忽略）。

- 影响：
  - 用户侧：Ideogram-4 推理 eager 模式 denoise 耗时 H100 -5.1%、H200 -4.7%，端到端 H200 5.211s → 4.898s，输出 md5 与 main 完全一致，属于零精度损失的纯性能优化，用户无需改任何配置。
  - 系统侧：新增的 silu_mul_bitexact.py 与 #34306 的 rope_rotate_half_bitexact.py 构成 diffusion 模块 bit-exact elementwise 融合工具集，后续其他 DiT 模型（如 ERNIE、未来新模型）可直接复用同一套 kernel 与 BitExactFusionGate 挂载模式。
  - 团队侧：确立了「融合 kernel + 首调自校验 + 永久回退」的安全优化流程，使位级敏感的优化可以默认开启而无需 A/B 开关；与 #34305、#34306 构成同一优化链条（权重反量化 → RoPE → SwiGLU 依次消解 eager/compile 差距）。
  - 风险标记：默认开启融合路径，位级一致性依赖数值实现，跨 PR 共享 kernel 文件，测试覆盖依赖 CUDA

关联脉络

- PR #34305 移除 per-forward FP8 权重反量化（标题未提供，PR body 引用）：本 PR 的优化起点：去掉 FP8 反量化后，Ideogram-4 剩余 eager 与 compile 的差距集中在 elementwise 链（约 15% 每步 GPU 时间）；两者正交（权重 vs 激活），本 PR 测量基于未合入 #34305 的 main。
- PR #34306 为 ERNIE 引入 rope_rotate_half_bitexact（标题未提供，PR body 引用）：本 PR 复用了同一份 rope_rotate_half_bitexact.py（内容一致，可任意顺序合并），并沿用其 BitExactFusionGate 挂载模式。
- PR #34401 Fix model-driven DiT layerwise offload auto policy: 同为 multimodal_gen diffusion 模块的性能 / 行为修复与重构，反映该模块近期在 eager 路径上的持续优化趋势。