PR #33886 完整报告

sgl-project/sglang

[diffusion] Z-Image bit-exact fused qk-norm (H200 Turbo 1024px e2e -6.4%)

合并时间：2026-08-07 17:03

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33886

执行摘要

- 一句话：Z-Image qk-norm 融合 Triton 内核，e2e 提速 6.4%
- 推荐动作：建议精读。核心看点：(1) 如何用 Triton 逐位复刻 aten 归约顺序（vec8 lane 串行累加 + shfl-down butterfly）并做到 torch.equal；(2) 融合过程中吸收 .contiguous() 物化的设计，把 q/k 拷贝开销消掉；(3) 运行时自检从引入到移除的权衡——先以进程级安全网验证可行性，再收敛为单元测试契约。维护时重点关注 bit-exact 契约对 aten 实现细节的依赖程度，以及测试用例是否足以在 aten 变更时及时暴露回归。

功能与动机

PR #29742 为修复 Z-Image 精度，在主注意力路径（precomputed rope cache）传 allow_inplace=False，把 qk-norm 强制到 eager ZImageRMSNorm 链。PR body 的 torch-profiler 显示该链是 denoise 步骤最贵的开销：eager qk-norm 链占 7.9%、q/k .contiguous() 拷贝占 3.7%、独立 flashinfer rope 启动占 1.0%。目标是在不改变任何数值输出的前提下，用融合 kernel 吸收这部分开销。

实现拆解

1. 瓶颈定位与融合范围选择：PR #29742 使主注意力路径走 eager ZImageRMSNorm 链，profiler 定位 qk-norm 链与 q/k 物化为主要开销。本 PR 只对最热的 rope-cache 主路径做 fused 优化，其他 rope 分支（freqs_cis）、量化运行和 torch.compile 场景保持原 eager 回退。
2. 新增 bit-exact Triton kernel：在 python/sglang/kernels/ops/diffusion/triton/zimage_native_norm.py 实现 _qk_rmsnorm_native_kernel，逐位复刻 aten 的数值 DAG —— bf16 行 128 场景下 16 字节向量化加载、lane 内串行 fp32 累加 8 个连续平方（平方先按 bf16 舍入）、shfl-down butterfly 归约、fp32 opmath 加 eps 后再按 bf16 舍入。同时新增 can_use_qk_rmsnorm_native 与 _qk_head_token_stride 做前置布局检查（4D 视图、head 块连续、token 均匀 stride、bf16、head_dim=128、CUDA），不满足即返回 None 走回退。
3. 模型侧接线：zimage.py 新增 zimage_native_qk_rmsnorm 封装；forward 中只在 qk_norm + rope_cos_sin_cache + 非 torch.compile + 开关开启 时尝试 fused 路径，且把 q/k 的 contiguous 物化推迟到 kernel 内完成（v 仍保留拷贝）。fused 成功后复用 apply_flashinfer_rope_qk_inplace，positions 语义与 eager fallback 一致。

4. 安全网设计与收敛：提交 1 引入进程级 torch.equal 运行时自检（首次调用不匹配则永久禁用 fused 并告警）；提交 2 移除该全局状态，改为由单元测试固定 bit-exact 契约，避免隐藏运行时开关。
5. 测试与验证配套：新增 test_zimage_qknorm_fusion.py，覆盖 strided fused-qkv 视图上的 bit-exact（torch.equal）与不支持的 head_dim 回退；PR body 给出 9M+ 行跨尺度验证、E2E PNG md5 四臂全等，以及 H200 性能数据（e2e -6.4%）。

关键文件：

- python/sglang/multimodal_gen/runtime/models/dits/zimage.py（模块 模型层；类别 source；类型 core-logic；符号 zimage_native_qk_rmsnorm）：Z-Image 主路径接线文件：新增 zimage_native_qk_rmsnorm 封装，并在 forward 的 rope-cache 分支引入 fused 路径与 eager fallback 的完整控制流，决定优化是否生效。
- python/sglang/kernels/ops/diffusion/triton/zimage_native_norm.py（模块 内核层；类别 source；类型 core-logic；符号 _qk_rmsnorm_native_kernel, _qk_head_token_stride, can_use_qk_rmsnorm_native, zimage_qk_rmsnorm_native）：PR 的核心：新增 bit-exact Triton kernel _qk_rmsnorm_native_kernel，逐位复刻 eager ZImageRMSNorm 的 aten 数值 DAG，并直接读取 strided fused-qkv 切片、写出 contiguous 结果，吸收 q/k 拷贝开销。
- python/sglang/multimodal_gen/test/unit/test_zimage_qknorm_fusion.py（模块 单元测试；类别 test；类型 test-coverage；符号 TestZImageQkNormFusion, _make_norm, test_bit_exact_on_strided_qkv_view, test_unsupported_head_dim_falls_back）：bit-exactness 的固定契约：用 torch.equal 校验 strided fused-qkv 视图上的融合结果与 eager ZImageRMSNorm 完全一致，并覆盖不支持 head_dim 时的回退行为。

关键符号：zimage_native_qk_rmsnorm, zimage_qk_rmsnorm_native, can_use_qk_rmsnorm_native, _qk_head_token_stride, _qk_rmsnorm_native_kernel, TestZImageQkNormFusion

关键源码片段

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

Z-Image 主路径接线文件：新增 zimage_native_qk_rmsnorm 封装，并在 forward 的 rope-cache 分支引入 fused 路径与 eager fallback 的完整控制流，决定优化是否生效。

```
def zimage_native_qk_rmsnorm(
    q: torch.Tensor,
    k: torch.Tensor,
    norm_q: ZImageRMSNorm,
    norm_k: ZImageRMSNorm,
    head_dim: int,
) -> tuple[torch.Tensor, torch.Tensor] | None:
    # Fused per-head ZImageRMSNorm for q/k, bit-exact vs the eager fallback.
    # 用一个 Triton launch 直接读 strided fused-qkv 切片，返回 contiguous (q, k)，
    # 不支持时返回 None 交给调用方走 eager 回退。
    from sglang.kernels.ops.diffusion.triton.zimage_native_norm import (
        can_use_qk_rmsnorm_native,
```

```
        zimage_qk_rmsnorm_native,
    )

    q_weight = norm_q.weight.data
    k_weight = norm_k.weight.data
    # 前置能力检查：任一输入不满足布局 / 精度条件就整体返回 None
    if not (
        can_use_qk_rmsnorm_native(q, q_weight, head_dim)
        and can_use_qk_rmsnorm_native(k, k_weight, head_dim)
    ):
        return None
    q_out = zimage_qk_rmsnorm_native(q, q_weight, norm_q.variance_epsilon)
    k_out = zimage_qk_rmsnorm_native(k, k_weight, norm_k.variance_epsilon)
    if q_out is None or k_out is None:
        return None
    return q_out, k_out


# ---- ZImageTransformerBlock.forward 中主路径接线（rope-cache 分支）----
# 只在最热路径（qk_norm + rope 缓存 + 非 torch.compile）尝试 fused，
# 量化和 torch.compile 等场景永远走原 eager 链
try_native_qk_norm = (
    self.qk_norm
    and rope_cos_sin_cache is not None
    and self.enable_zimage_qk_fusion
    and not torch.compiler.is_compiling()
)
if self.use_fused_qkv:
    qkv, _ = self.to_qkv(hidden_states)
    q, k, v = qkv.split(
        [
            self.local_num_heads * self.head_dim,
            self.local_num_kv_heads * self.head_dim,
            self.local_num_kv_heads * self.head_dim,
        ],
        dim=-1,
    )
    # q/k 的 contiguous 物化推迟给 fused kernel；否则仍按原逻辑提前拷贝
    if not try_native_qk_norm:
        q = q.contiguous()
        k = k.contiguous()
    v = v.contiguous()
# ...
if rope_cos_sin_cache is not None:
    if self.qk_norm:
        fused_qk = None
        if try_native_qk_norm:
            fused_qk = zimage_native_qk_rmsnorm(
                q, k, self.norm_q, self.norm_k, self.head_dim
```

```
            )
        if fused_qk is not None:
            q, k = fused_qk
            # positions=None 时与 eager fallback 语义一致：按 seqlen arange 并跨 batch 重复
            q, k = apply_flashinfer_rope_qk_inplace(
                q,
                k,
                rope_cos_sin_cache,
                head_size=self.head_dim,
                is_neox=False,
                positions=rope_positions,
            )
        else:
            q = q.contiguous()
            k = k.contiguous()
            q, k = apply_qk_norm_with_optional_rope(
                q=q, k=k, q_norm=self.norm_q, k_norm=self.norm_k,
                head_dim=self.head_dim, cos_sin_cache=rope_cos_sin_cache,
                is_neox=False, positions=rope_positions,
                allow_inplace=False, # 保持 #29742 引入的原生 bf16 数值轨迹
            )
```

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

PR 的核心：新增 bit-exact Triton kernel _qk_rmsnorm_native_kernel，逐位复刻 eager ZImageRMSNorm 的 aten 数值 DAG，并直接读取 strided fused-qkv 切片、写出 contiguous 结果，吸收 q/k 拷贝开销。

```
@triton.jit
def _qk_rmsnorm_native_kernel(
    y_ptr,
    x_ptr,
    weight_ptr,
    token_stride,
    nheads,
    n_rows,
    head_dim: tl.constexpr,
    eps: tl.constexpr,
    rows_per_prog: tl.constexpr,
):
    # Per-head RMSNorm replicating the eager ZImageRMSNorm kernel chain bit-for-bit.
    # 复刻要点：bf16 行 128 时 aten reduce_kernel 用 16 字节向量化加载，
    # 每个 lane 串行累加 8 个连续平方（平方先按 bf16 舍入），再做 shfl-down
    # butterfly 归约；所有中间结果在与 aten 相同的边界点按 bf16 舍入。
    prog = tl.program_id(0)
    row_offs = tl.arange(0, rows_per_prog)
    rows = prog * rows_per_prog + row_offs
    row_mask = rows < n_rows
    # 把扁平行号 (token, head) 拆开，直接按 fused-qkv 的 stride 取地址
    tokens = rows // nheads
```

```
    heads = rows % nheads
    base = x_ptr + tokens * token_stride + heads * head_dim

    offs = tl.arange(0, head_dim)
    x = tl.load(base[:, None] + offs[None, :], mask=row_mask[:, None], other=0.0)
    # aten::pow 的平方结果先舍入到 bf16，再进入 fp32 累加
    sq = (x * x).to(tl.bfloat16)

    # 用有序 tl.split 链把 8 元素组按 lane 串行顺序拆开，
    # 保证 fp32 累加次序与 aten 完全一致（s0+...+s7）
    g = tl.reshape(sq, (rows_per_prog, head_dim // 8, 4, 2), can_reorder=False)
    p0, p1 = tl.split(g)
    p00, p01 = tl.split(tl.reshape(p0, (rows_per_prog, head_dim // 8, 2, 2), can_reorder=False))
    p10, p11 = tl.split(tl.reshape(p1, (rows_per_prog, head_dim // 8, 2, 2), can_reorder=False))
    s0, s4 = tl.split(tl.reshape(p00, (rows_per_prog, head_dim // 8, 1, 2), can_reorder=False))
    s2, s6 = tl.split(tl.reshape(p01, (rows_per_prog, head_dim // 8, 1, 2), can_reorder=False))
    s1, s5 = tl.split(tl.reshape(p10, (rows_per_prog, head_dim // 8, 1, 2), can_reorder=False))
    s3, s7 = tl.split(tl.reshape(p11, (rows_per_prog, head_dim // 8, 1, 2), can_reorder=False))
    acc = tl.reshape(s0, (rows_per_prog, head_dim // 8)).to(tl.float32)
    acc = acc + tl.reshape(s1, (rows_per_prog, head_dim // 8)).to(tl.float32)
    acc = acc + tl.reshape(s2, (rows_per_prog, head_dim // 8)).to(tl.float32)
    acc = acc + tl.reshape(s3, (rows_per_prog, head_dim // 8)).to(tl.float32)
    acc = acc + tl.reshape(s4, (rows_per_prog, head_dim // 8)).to(tl.float32)
    acc = acc + tl.reshape(s5, (rows_per_prog, head_dim // 8)).to(tl.float32)
    acc = acc + tl.reshape(s6, (rows_per_prog, head_dim // 8)).to(tl.float32)
    acc = acc + tl.reshape(s7, (rows_per_prog, head_dim // 8)).to(tl.float32)

    # shfl-down butterfly（half = 8/4/2/1 的两两有序折叠，等价 aten 的 warp 归约）
    acc = tl.sum(tl.reshape(acc, (rows_per_prog, 2, 8), can_reorder=False), axis=1)
    acc = tl.sum(tl.reshape(acc, (rows_per_prog, 2, 4), can_reorder=False), axis=1)
    acc = tl.sum(tl.reshape(acc, (rows_per_prog, 2, 2), can_reorder=False), axis=1)
    ssum = tl.sum(acc, axis=1)
    ms = (ssum / head_dim).to(tl.bfloat16)
    # aten 在 fp32 opmath 上加 eps，rsqrt 结果先舍入到 bf16 再参与后续乘法
    rstd = tl.rsqrt((ms.to(tl.float32) + eps).to(tl.bfloat16).to(tl.float32)).to(tl.bfloat16)

    weight = tl.load(weight_ptr + offs)
    # x * rstd 的乘积累积 bf16 舍入点与 eager mul 一致，再乘 weight
    y = ((x.to(tl.float32) * rstd.to(tl.float32)[:, None]).to(tl.bfloat16) * weight).to(tl.bfloat16)
    tl.store(y_ptr + rows[:, None] * head_dim + offs[None, :], y, mask=row_mask[:, None])
```

评论区精华

该 PR 无公开 review 评论（review 评论为 0），唯一 issue 评论是作者 BBuf 贴的 CI 链接。最有价值的设计讨论体现在两次提交的演进中：提交 1 加入了进程级 torch.equal 运行时自检作为安全网（不匹配则永久禁用 fused 路径），提交 2（5c9e7df）将其移除，提交消息写道：“Drop the global self-check ... bit-exactness is pinned by the unit test instead of a hidden runtime switch”。这是一个值得注意的权衡：用显式单元测试契约替代进程内隐藏开关

，消除运行时开销与全局副作用，代价是数值契约对 aten 实现变化的防护从运行时变成了 CI 时。

- 运行时 bit-exact 自检的去留 (design): 移除隐藏运行时开关与进程级全局状态，数值契约由 test_zimage_qknorm_fusion.py 固定；kernel 或 aten 行为变化时靠 CI 暴露。

风险与影响

- 风险：
  - 数值契约对 aten 实现的耦合：_qk_rmsnorm_native_kernel 精确复刻 aten reduce_kernel 在 bf16、head_dim=128 下的 16 字节向量化加载、lane 串行 fp32 累加与 shfl-down butterfly 顺序。若未来 aten dispatch、向量化宽度或 GPU 平台行为变化，bit-exact 可能被破坏；提交 2 移除运行时自检后，仅靠 test_zimage_qknorm_fusion.py 两个用例把关，覆盖规模有限（随机数据、非真实模型分布）。
  - 适用范围窄但 fallback 完整：仅 CUDA + bf16 + head_dim=128 + 4D 且 head 块连续、token 均匀 stride 的输入走 fused；量化、torch.compile、freqs_cis 分支等仍走 eager。前置检查 can_use_qk_rmsnorm_native 返回 None 即回退，行为不变。
  - 物化语义变化：q/k 不再提前 contiguous()，而是由 kernel 直接读 strided 切片并输出 contiguous 张量；v 仍保留拷贝。需关注后续 apply_flashinfer_rope_qk_inplace 与注意力后端对 q/k 布局的假设（本 PR 已在 E2E 验证）。
  - 性能收益的普适性：-6.4% 是 H200 + 1024x1024 + Turbo 9 步 preset 的结果，其他分辨率、步数、GPU 型号的收益需要另行验证。
  - 影响：影响范围集中于 sglang/multimodal_gen 的 Z-Image / Z-Image-Turbo 推理：默认 eager 配置下 rope-cache 主路径性能提升约 6%，且输出与之前逐位一致，用户无感知迁移成本。对量化、torch.compile、非常规布局用户完全透明（走原 eager 路径）。对团队而言，该 PR 提供了一种可复用的"bit-exact 融合 eager aten 链"模式：前置能力检查 + None 回退 + 测试固定契约，为后续吸收其他模型的规范化 kernel 链（如 MiniMax-H3、GLM-Image 的 RMSNorm 链）提供参考；同时新增一个需要与 aten 行为保持同步的 Triton kernel，带来长期维护成本。CI 新增一个 CUDA-dependent 单元测试（非 CUDA 环境 skip）。
  - 风险标记：bit-exact 依赖 aten 数值实现，运行时自检已移除，仅 bf16/head_dim=128/CUDA 生效，torch.compile/ 量化走回退路径

关联脉络

- PR #29742 #29742 修复 Z-Image 精度（标题未在上下文中提供）：PR body 明确引用：allow_inplace=False 强制 eager qk-norm 链，是本 PR 的性能瓶颈来源；本 PR 在其数值轨迹之上做 bit-exact 融合。
- PR #33923 [Diffusion] Route zimage and hunyuanvideo attention through USPAttention: 同样修改 zimage.py 的注意力路径，将 zimage 注意力迁入 USPAttention，与本 PR 同属 Z-Image 注意力链路演进线。

- PR #32667 [Diffusion] Add K/V-gather sequence parallel attention: 为 Diffusion 新增 K/V-gather 序列并行注意力并在 SP2 默认启用，同样涉及 zimage.py 注意力路径与 SP 布局，与本 PR 改动区域相邻。