PR #36504 完整报告

sgl-project/sglang

[diffusion][kernel] support transposed residual-gate add

合并时间：2026-08-28 16:54

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/36504

执行摘要

- 一句话：支持 SANA-Video 转置残差门控，内核提速 2.3 倍
- 推荐动作：值得精读。该 PR 展示了一种在 CUDA kernel 内部消化非标准张量布局、避免数据搬运的典型技巧，适合作为 diffusion 内核优化的参考案例。关注 _is_transposed_dense_residual 的守卫设计、共享内存 tile 的 padding 策略，以及 torch.empty_strided 在 CUDA Graph 场景下的用法。

功能与动机

PR body 中给出 H100 上的 profile 证据：SANA-Video 每个 transformer block 的两个 gated residual 站点未命中既有连续布局融合，residual 的实际 stride 为 (17472000, 1, 7800)（逻辑 [1, 7800, 2240]，底层 [1, 2240, 7800] 连续）。trace 显示 106.1 ms 分布在 720 个 BF16 add 启动上，若将 residual 物化为连续需要额外整张拷贝，因此选择在 kernel 内直接消化转置布局。

实现拆解

1. 布局识别与守卫：在 python/sglang/kernels/ops/diffusion/modulate/residual_gate_add_jit.py 中新增 _is_transposed_dense_residual()，精确匹配 [B, tokens, hidden] 逻辑形状 + stride (tokens * hidden, 1, tokens)，并要求 update/gate 连续、gate 为 [1, 1, hidden] 行广播、三维 tile 化网格不超过 CUDA 65535 上限。can_use_residual_gate_add_cuda() 的守卫条件扩展为“连续输入或满足转置条件”，不支持的 dtype/layout 仍走 eager fallback。
2. JIT 分支与输出布局保持：在 _residual_gate_add_custom_op() 中，命中转置路径时调用新注册的 residual_gate_add_transposed 符号；输出改用 torch.empty_strided(residual.shape, residual.stride()) 分配，从而原样保留输入的 stride，天然满足 CUDA Graph 捕获的布局稳定性要求。
3. CUDA 转置内核：在 python/sglang/kernels/jit/csrc/diffusion/residual_gate_add.cuh 中新增 residual_gate_add_transposed_kernel。每个 block 处理 32 x 32 tile：先用 [32][33]（带 padding）共享内存按逻辑行主序协作加载 update 并保证合并读，再 __syncthreads() 后按转置索引从共享内存消费，使 residual 读与 out 写都落在底层 [B, hidden, tokens] 的连续地址上。ResidualGateAddKernel 新增 run_transposed 静态入口，通过 TensorView 匹配 [B, S, D] 与转置 stride。

4. 测试与基准：test_modulate.py 新增 4 个测试：转置形状（dtype × shape 参数化）、非零 storage offset、torch.compile(fullgraph=True)、CUDA Graph 捕获 / 回放，并断言输出 stride 与参考一致。bench_residual_gate_add.py 新增 SANA-Video 生产形状的转置 workload，CI 中用小 shape，非 CI 用 [1, 7800, 2240]；输出表格增加 reference 列，转置场景以 PyTorch 为基准。
5. 文档：kernels/ops/diffusion/README.md 增加 “Residual gating” 表格条目，说明转置路径约定和 tile 机制；.claude/skills/.../existing-fast-paths.md 将第 10 条快速路径扩展为共享残差门控融合（LTX2、LongCat-Image、SANA、SANA-Video），并强调不要为触达普通路径插入 .contiguous()。

关键文件：

- python/sglang/kernels/jit/csrc/diffusion/residual_gate_add.cuh（模块 CUDA 内核；类别 source；类型 core-logic；符号 residual_gate_add_transposed_kernel, run_transposed）：核心 CUDA 内核，新增 32 x 32 共享内存 tile 的转置实现，是本次性能提升的关键。
- python/sglang/kernels/ops/diffusion/modulate/residual_gate_add_jit.py（模块 JIT 调度；类别 source；类型 core-logic；符号 _is_transposed_dense_residual, _TRANSPOSE_TILE, _MAX_GRID_DIM）：新增转置布局识别守卫与 JIT 分派，决定何时走转置内核并保持输出 stride。
- test/registered/kernels/ops/diffusion/test_modulate.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_residual_gate_add_transposed_residual, test_residual_gate_add_transposed_storage_offsets, test_residual_gate_add_transposed_torch_compile_fullgraph, test_residual_gate_add_transposed_cuda_graph）：新增 4 个针对性测试，覆盖转置形状、storage offset、torch.compile fullgraph 与 CUDA Graph，验证正确性与布局保持。
- test/registered/kernels/benchmark/diffusion/bench_residual_gate_add.py（模块 性能基准；类别 test；类型 test-coverage）：新增 SANA-Video 转置 workload，并将转置场景的基准参照切换为 PyTorch，保证基准可度量。
- python/sglang/kernels/ops/diffusion/README.md（模块 文档；类别 docs；类型 documentation）：补充转置路径的契约说明与设计动机，帮助开发者理解为何不应插入 contiguous。
- python/sglang/multimodal_gen/.claude/skills/sglang-diffusion-benchmark-profile/existing-fast-paths.md（模块 文档；类别 docs；类型 documentation）：更新快速路径文档，将 SANA/SANA-Video 纳入共享残差门控融合范围并强调约束。

关键符号：residual_gate_add_transposed_kernel, ResidualGateAddKernel::run_transposed, _is_transposed_dense_residual, _residual_gate_add_custom_op, test_residual_gate_add_transposed_residual, test_residual_gate_add_transposed_storage_offsets, test_residual_gate_add_transposed_torch_compile_fullgraph, test_residual_gate_add_transposed_cuda_graph

关键源码片段

python/sglang/kernels/jit/csrc/diffusion/residual_gate_add.cuh

核心 CUDA 内核，新增 32 x 32 共享内存 tile 的转置实现，是本次性能提升的关键。

```
// 转置残差门控加法内核：residual/output 以 [B, hidden, tokens] 连续布局存储，
// 但以 [B, tokens, hidden] 逻辑形状暴露；update 则以逻辑行主序连续存储。
template <typename T>
__global__ void residual_gate_add_transposed_kernel(
    T* __restrict__ out,
    const T* __restrict__ residual,
    const T* __restrict__ update,
    const T* __restrict__ gate,
    int64_t tokens,
    int64_t hidden_size) {
  // 32 x 32 共享内存 tile，每行多 1 个元素 padding，避免 bank conflict
  __shared__ T update_tile[kTransposeTile][kTransposeTile + 1];

  const int64_t batch = blockIdx.z;
  const int64_t token_base = static_cast<int64_t>(blockIdx.x) * kTransposeTile;
  const int64_t hidden_base = static_cast<int64_t>(blockIdx.y) * kTransposeTile;
  const int64_t batch_offset = batch * tokens * hidden_size;

  // 第一阶段：以逻辑行主序协作加载 update tile，保证读取合并
#pragma unroll
  for (uint32_t item = threadIdx.x; item < kTransposeTile * kTransposeTile;
       item += kTransposeBlockSize) {
    const uint32_t token_in_tile = item / kTransposeTile;
    const uint32_t hidden_in_tile = item % kTransposeTile;
    const int64_t token = token_base + token_in_tile;
    const int64_t hidden = hidden_base + hidden_in_tile;
    if (token < tokens && hidden < hidden_size) {
      update_tile[token_in_tile][hidden_in_tile] =
          update[batch_offset + token * hidden_size + hidden];
    }
  }
  __syncthreads();

  // 第二阶段：以转置顺序消费共享内存，残差 / 输出读写按 [B, hidden, tokens] 连续
#pragma unroll
  for (uint32_t item = threadIdx.x; item < kTransposeTile * kTransposeTile;
       item += kTransposeBlockSize) {
    const uint32_t hidden_in_tile = item / kTransposeTile;
    const uint32_t token_in_tile = item % kTransposeTile;
    const int64_t token = token_base + token_in_tile;
    const int64_t hidden = hidden_base + hidden_in_tile;
    if (token < tokens && hidden < hidden_size) {
      const int64_t transposed_offset = batch_offset + hidden * tokens + token;
      out[transposed_offset] = residual_gate_value(
          residual[transposed_offset],
          update_tile[token_in_tile][hidden_in_tile],
```

```
          SGLANG_LDG(gate + hidden));
    }
  }
}
```

python/sglang/kernels/ops/diffusion/modulate/residual_gate_add_jit.py

新增转置布局识别守卫与 JIT 分派，决定何时走转置内核并保持输出 stride。

```
# 识别 SANA-Video 使用的转置稠密布局：
# residual/output 在内存中是 [B, hidden, tokens] 连续布局，但以
# [B, tokens, hidden] 逻辑形状和 stride (tokens * hidden, 1, tokens) 暴露。
# 只要 update/gate 为逻辑连续且形状满足约束，就走专属 transposed kernel，
# 避免为了访问普通路径而物化连续副本（那会带来整张张量的拷贝开销）。
def _is_transposed_dense_residual(
    residual: torch.Tensor, update: torch.Tensor, gate: torch.Tensor
) -> bool:
    # 仅支持 3D 张量 + [1, 1, hidden] 行广播 gate
    if residual.dim() != 3 or gate.shape != (1, 1, residual.shape[-1]):
        return False
    batch, tokens, hidden_size = residual.shape

    # 按 32 x 32 tile 划分后的 grid 尺寸必须落在 CUDA 的 65535 限制内
    return (
        batch <= _MAX_GRID_DIM
        and (tokens + _TRANSPOSE_TILE - 1) // _TRANSPOSE_TILE <= _MAX_GRID_DIM
        and (hidden_size + _TRANSPOSE_TILE - 1) // _TRANSPOSE_TILE <= _MAX_GRID_DIM
        # stride 精确匹配转置稠密：内存中行优先为 [hidden, tokens]
        and residual.stride() == (tokens * hidden_size, 1, tokens)
        and update.is_contiguous()
        and gate.is_contiguous()
    )


# 自定义 op 入口：先判断是否命中转置专用路径，否则走普通连续路径
def _residual_gate_add_custom_op(
    residual: torch.Tensor, update: torch.Tensor, gate: torch.Tensor
) -> torch.Tensor:
    # 用 empty_strided 保留输入 stride，避免 CUDA graph 捕获时改变输出布局
    out = torch.empty_strided(
        residual.shape,
        residual.stride(),
        dtype=residual.dtype,
        device=residual.device,
    )
    module = _jit_residual_gate_add_module(residual.dtype)
    if _is_transposed_dense_residual(residual, update, gate):
        module.residual_gate_add_transposed(out, residual, update, gate)
        return out
    # 普通路径：gate 可能为全量或行广播，此处省略非转置分支的细节
```

```
broadcast_gate = gate.shape != residual.shape
# ... 后续调用 module.residual_gate_add(...)
```

评论区精华

该 PR 没有 review 讨论线程。PR 作者在正文中明确说明了设计权衡：使用 32 x 32 padded 共享内存 tile 使 update 读取与 residual/output 流量保持合并，而非插入整张 .contiguous() 拷贝。源码注释中也强调 CUDA Graph 需在捕获前完成一次 JIT 编译与 warmup，避免捕获阶段产生额外分配。

- 暂无高价值评论线程

风险与影响

- 风险：
  - 网格维度限制：转置内核使用 z 方向表示 batch、x/y 方向表示 token/hidden 的 tile，_is_transposed_dense_residual 对三个方向做 <= 65535 检查，异常大的 batch 或 hidden 会退回 *eager*，属于安全降级。
  - stride 假设：转置路径只在 residual.stride() == (tokens * hidden, 1, tokens) 时启用，未来若模型产生不同视图（如非满 batch slice）会走普通 fallback，不影响正确性但可能错过优化。
  - CUDA Graph：torch.empty_strided 与 JIT 模块在捕获前已 warmup，测试覆盖 capture/replay；但若模型 warmup 画面与请求 shape 不一致（PR body 提到 BCG 签名差异），端到端路径可能不捕获该自定义 op，收益会缩小。
  - AMD ROCm CI在该 PR 上失败（Run #33082942491），但失败原因与本次内核改动是否相关尚不明确，需关注后续验证。
  - 影响：对 SANA-Video 用户而言，8 步 832x480x17 推理端到端约提升 2.8%~4.2%，内核微基准提速 2.291 倍。对框架开发者而言，共享 residual_gate_add 现在支持连续与转置两种布局，后续其他 diffusion 模型可复用该模式。团队维护上，新增了 JIT 符号与守卫逻辑，需要保证文档与基准同步更新。
  - 风险标记：GPU 内核新增转置路径，strict stride guard, AMD ROCm CI 失败，端到端收益受 warmup 签名影响

关联脉络

- PR #36521 [diffusion][kernel] avoid 4D scale-shift autotuning: 同属 diffusion JIT 内核性能优化系列，改动了同一目录的 modulate 与 benchmark 文件，且同样强调减少冷启动开销。
- PR #36658 [multimodal_gen] fix: make tail_attn_meta CUDA-graph capturable: 涉及 CUDA Graph 捕获正确性，与本次转置路径的 CUDA Graph 测试互相关联，共同保障 diffusion 后端的图捕获稳定性。
- PR #36726 [Diffusion] Fix the five unit tests failing on main: 修复了 diffusion 相关单元测试，与本次新增测试同属于 diffusion 测试体系，可能影响 CI 基线。