

PR #38042 完整报告

sgl-project/sglang

[diffusion] add Helios per-token gated-residual fusion (quality-gated)

合并时间: 2026-09-05 10:29

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/38042>

执行摘要

- 一句话: 为 Helios 视频 DiT 模型新增质量门控的残差 + 门控融合内核, 消除分散的逐元素计算与类型转换, 提升推理性能。
- 推荐动作: 该 PR 值得精读, 尤其关注: 1) 融合站点如何通过 QualityGatedFusion 抽象实现请求级开关; 2) HeliosTransformerBlock.forward 中融合路径与参考路径的条件切换逻辑; 3) HeliosChunkedDenoisingStage._maybe_toggle_quality_fusions 如何与现有的质量参数体系集成。设计上遵循了仓库已有的融合模式 (如 LingBot), 代码改动集中且目标明确, 是一个典型的、高质量的性能优化 PR。

功能与动机

基于性能分析, 在 B300 GPU 上运行 BestWishYsh/Helios-Distilled 模型时, Helios transformer block 中分散的 `residual + (gate * update).to(dtype)` 操作链 (FP32 计算后转换为 BF16) 占用了大量 GPU 时间 (PR body 指出约 18% 在残差加法, 17% 在类型拷贝)。现有 `residual_gate_add` 内核已能高效计算 `residual + update * gate`, 但 Helios 从未使用。此外, Helios 运行在专用的 `HeliosChunkedDenoisingStage`, 而非标准 `DenoisingStage`, 导致其从未启用过请求级的质量融合。PR 的目标是复用该内核以消除性能瓶颈, 并将 Helios 纳入标准的质量融合框架。

实现拆解

1. 新增融合站点基础设施: 创建 `python/sglang/kernels/ops/diffusion/sites/helios_gated_residual_site.py`, 定义 `QualityGatedFusion` 实例, 并提供 `mark_helios_gated_residual_site`、`mount_helios_gated_residual`、`unmount_helios_gated_residual` 和核心函数 `try_helios_gated_residual`。此文件是融合逻辑的实现核心。
2. 修改模型核心前向路径: 在 `python/sglang/multimodal_gen/runtime/models/dits/helios.py` 的 `HeliosTransformerBlock.__init__` 中调用 `mark_helios_gated_residual_site(self)`, 在 `forward` 方法中, 用 `try_helios_gated_residual` 替换原有的两处残差计算 (自注意力后和 FFN 后)。当融合激活时, 直接使用 `residual_gate_add` 内核; 否则保留原始的 FP32 乘加路径。
3. 集成质量门控调度: 在 `python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/helios_denoising.py` 的 `HeliosChunkedDenoisingStage` 中, 新增 `_maybe_toggle_quality_fusions` 方法。该方法在 `forward` 入口处根据请求的 `quality` 参数

调用 `mount_helios_gated_residual` 或 `unmount_helios_gated_residual`，实现请求级的融合开关。

4. 注册至全局融合处理器：在 `python/sglang/multimodal_gen/runtime/pipelines_core/stages/denoising.py` 的 `_QUALITY_FUSION_HANDLERS` 元组中添加新条目 ("Helios per-token gated residual", `mount_helios_gated_residual`, `unmount_helios_gated_residual`)，使该融合可通过标准流程管理。
5. 更新模块导出：在 `python/sglang/kernels/ops/diffusion/__init__.py` 的 `__all__` 字典中添加新的符号 (`helios_gated_residual_active`, `mark_helios_gated_residual_site`, `mount_helios_gated_residual`, `try_helios_gated_residual`, `unmount_helios_gated_residual`)，使其可通过 `sglang.kernels.ops.diffusion` 访问。本次变更无配套测试文件修改。

关键文件：

- `python/sglang/kernels/ops/diffusion/sites/helios_gated_residual_site.py`（模块 融合内核；类别 `infra`；类型 `infrastructure`；符号 `mark_helios_gated_residual_site`, `helios_gated_residual_active`, `mount_helios_gated_residual`, `unmount_helios_gated_residual`）：新增的核心文件，实现了 Helios 门控残差融合的完整逻辑，包括标记、激活检查、挂载 / 卸载和尝试融合的函数。
- `python/sglang/multimodal_gen/runtime/models/dits/helios.py`（模块 模型定义；类别 `source`；类型 `data-contract`；符号 `HeliosTransformerBlock.init`, `HeliosTransformerBlock.forward`）：模型核心文件，修改了 `HeliosTransformerBlock` 的前向计算，在 `__init__` 中标记融合站点，在 `forward` 中用 `try_helios_gated_residual` 替换两处原始计算。
- `python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/helios_denoising.py`（模块 去噪阶段；类别 `source`；类型 `data-contract`；符号 `HeliosChunkedDenoisingStage._maybe_toggle_quality_fusions`, `HeliosChunkedDenoisingStage.forward`）：Helios 专用去噪阶段，新增了 `_maybe_toggle_quality_fusions` 方法，在 `forward` 中调用以根据请求质量参数挂载 / 卸载融合。
- `python/sglang/multimodal_gen/runtime/pipelines_core/stages/denoising.py`（模块 融合注册；类别 `source`；类型 `core-logic`；符号 `_QUALITY_FUSION_HANDLERS`）：标准去噪阶段，将新的 Helios 融合处理器注册到全局 `_QUALITY_FUSION_HANDLERS` 列表中，使其能被标准流程管理。
- `python/sglang/kernels/ops/diffusion/__init__.py`（模块 内核导出；类别 `config`；类型 `data-contract`；符号 `all`）：内核操作的公共导出文件，将新的 Helios 融合相关符号添加到 `__all__` 字典，使其可通过 `sglang.kernels.ops.diffusion` 访问。

关键符号：`mark_helios_gated_residual_site`, `helios_gated_residual_active`, `mount_helios_gated_residual`, `unmount_helios_gated_residual`, `try_helios_gated_residual`, `HeliosChunkedDenoisingStage._maybe_toggle_quality_fusions`

关键源码片段

python/sglang/kernels/ops/diffusion/sites/helios_gated_residual_site.py

新增的核心文件，实现了 Helios 门控残差融合的完整逻辑，包括标记、激活检查、挂载 / 卸载和尝试融合的函数。

```
"""Helios per-token gated-residual fusion, gated by request quality.
```

```
Each Helios block applies ``residual + (gate * update).to(residual.dtype)`` at
the self-attention and FFN updates, where ``update`` (post-attn / post-FFN) and
the per-token ``gate`` (``[B, S, 1]``) stay in FP32 while ``residual`` is BF16.
The shared ``residual_gate_add`` kernel computes ``residual + update * gate``
in a single pass but requires one dtype, so the gate and update are first cast
to BF16. That reordering of the FP32 multiply is numerically equivalent only at
half-precision rounding level (not bit-exact), so the fusion is opt-in:
``quality="extra-high"`` and ``quality="high"`` mount it, while the default
``quality="lossless"`` keeps the reference FP32-multiply form bit-for-bit.
"""
```

```
from __future__ import annotations
```

```
import logging
```

```
import torch
```

```
import torch.nn as nn
```

```
# 引入已有的 QualityGatedFusion 抽象，用于管理融合站点的挂载状态和启用标志
from sglang.kernels.ops.diffusion.sites.quality_gate import QualityGatedFusion
```

```
logger = logging.getLogger(__name__)
```

```
# 创建专属于 Helios 的融合配置实例，定义标记属性、启用属性和名称
```

```
_FUSION = QualityGatedFusion(
    name="Helios per-token gated residual",
    marker_attr="_sgl_helios_gated_residual_site",
    enabled_attr="_sgl_helios_gated_residual_enabled",
)
```

```
def mark_helios_gated_residual_site(module: nn.Module) -> None:
```

```
    """标记一个 Helios block；初始状态为参考路径（未激活）。"""
    _FUSION.mark(module)
```

```
def helios_gated_residual_active(module: nn.Module) -> bool:
```

```
    # 检查指定模块上的融合是否处于激活状态
    return _FUSION.is_enabled(module)
```

```
def mount_helios_gated_residual(root: nn.Module) -> bool:
```

```

# 在模型树中挂载融合，将根模块下所有标记的站点激活
return _FUSION.mount(root, logger=logger)

def unmount_helios_gated_residual(root: nn.Module) -> None:
    # 在模型树中卸载融合，将根模块下所有标记的站点停用
    _FUSION.unmount(root)

def try_helios_gated_residual(
    site: nn.Module,
    residual: torch.Tensor,
    update: torch.Tensor,
    gate: torch.Tensor,
) -> torch.Tensor | None:
    """尝试返回融合后的 ``residual + update * gate``。

    当融合站点未激活或张量不满足内核要求时返回 None（调用者执行参考路径）。
    """
    if not _FUSION.is_enabled(site):
        return None
    # 延迟导入，避免循环依赖，并仅在需要时检查 CUDA 内核可用性
    from sglang.kernels.ops.diffusion import (
        can_use_residual_gate_add_cuda,
        residual_gate_add,
    )

    # 确保 gate 和 update 与 residual 同类型（内核要求）
    if residual.dtype != update.dtype or residual.dtype != gate.dtype:
        update = update.to(residual.dtype)
        gate = gate.to(residual.dtype)
    # 检查张量形状和设备是否满足内核要求（例如门控为 [B,S,1]）
    if not can_use_residual_gate_add_cuda(residual, update, gate):
        return None
    # 调用已有的融合内核计算
    return residual_gate_add(residual, update, gate)

```

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

模型核心文件，修改了 HeliosTransformerBlock 的前向计算，在 __init__ 中标记融合站点，在 forward 中用 try_helios_gated_residual 替换两处原始计算。

```

class HeliosTransformerBlock(nn.Module):
    # ... (省略 __init__ 部分代码)

    def __init__(self, ...):
        # ... (其他初始化代码)
        # 4. Guidance cross-attention flag
        self.guidance_cross_attn = guidance_cross_attn

```

```

# 标记此 block 为融合站点，初始状态为参考路径
mark_helios_gated_residual_site(self)

def forward(
    self,
    hidden_states,
    encoder_hidden_states,
    temb,
    rotary_emb,
    original_context_length=None,
    cross_attn_key_value=None,
):
    # ... (temb 处理部分省略)

    # 1. Self-attention
    norm_hidden_states = self.norm1(hidden_states, shift_msa, scale_msa)
    attn_output = self.attn1(
        norm_hidden_states, rotary_emb, original_context_length
    )
    # 尝试使用融合内核计算残差加法
    fused = try_helios_gated_residual(self, hidden_states, attn_output, gate_msa)
    hidden_states = (
        fused
        if fused is not None
        else (hidden_states.float() + attn_output * gate_msa).type_as(hidden_states)
    )

    # 2. Cross-attention ( 省略 )
    # 3. Feed-forward
    norm_hidden_states = self.norm3(hidden_states, c_shift_msa, c_scale_msa)
    ff_output = self.ffn(norm_hidden_states)
    # 再次尝试融合
    fused = try_helios_gated_residual(self, hidden_states, ff_output, c_gate_msa)
    hidden_states = (
        fused
        if fused is not None
        else (hidden_states.float() + ff_output.float() * c_gate_msa).type_as(
            hidden_states
        )
    )

    return hidden_states

```

评论区精华

无正式 review 讨论记录。PR body 中详细阐述了设计决策和性能 / 精度权衡。关键决策是采用与 **LingBot** 类似的质量门控模式，确保数值精度可控：**quality="lossless"** 保持位精确参考路径，**high/extra-high** 启用可能引入 BF16 舍入误差的融合路径。PR 作者也指出 Helios 生成本

身具有非确定性，因此无法通过固定种子的 MD5 校验等同性。

- 性能分析与优化目标 (performance): 通过复用现有 residual_gate_add 内核进行融合，实测在 B300 上获得约 6.4% 的推理加速。
- 数值精度与质量门控 (design): 采用 opt-in 策略，平衡性能与精度，用户可控。
- Helios 专用阶段的集成 (design): 修复了集成遗漏，使 Helios 也能受益于质量融合框架。

风险与影响

• 风险:

1. 数值精度风险: 融合路径将 FP32 乘法结果重排序到 BF16 进行加法，是数值近似而非精确等价。虽然已通过 quality 门控将影响限制在用户明确选择高性能模式时，但仍可能在下流产生微小差异。由于 Helios 生成本身的非确定性，这种差异可能难以隔离和复现。
2. 融合内核兼容性风险: try_helios_gated_residual 内部依赖 can_use_residual_gate_add_cuda 检查，该检查需确保输入张量形状（如 [B, S, 1] 的门控）满足内核要求。若检查不完善或未来模型结构变化，可能导致意外回退到参考路径。
3. 线程安全与状态管理风险: 融合的挂载 / 卸载状态 (_sgl_helios_gated_residual_enabled) 存储在模块属性上，并通过 _maybe_toggle_quality_fusions 管理。虽然该逻辑在 stage 的 forward 中同步调用，但需确保在并发请求下不会因状态交叉导致融合行为异常。

• 影响:

1. 用户影响: 对于使用 Helios 蒸馏模型 (BestWishYsh/Helios-Distilled) 并设置 quality=high 或 extra-high 的用户，推理速度将提升约 6.4%（基于 B300 单卡测量）。默认 quality=lossless 不受影响，保证向后兼容。
2. 系统影响: 扩展了 SGLang 扩散模型的融合内核框架，为 Helios 添加了第一个性能优化融合点。增强了 quality 参数对内核选择的控制粒度。融合逻辑的实现复用了现有 QualityGatedFusion 基础设施，模式清晰，易于未来扩展。
3. 团队影响: 提供了一个将已优化内核适配到新模型的完整范例，涉及内核站点、模型前向改造和请求级调度三个层面。对其他 DiT 模型（如后续的 Helios 变体）的性能优化具有参考价值。 - 风险标记: 数值精度近似，内核兼容性检查，请求级状态管理

关联脉络

- PR #38001 [diffusion] auto-keep video DiT resident on high-memory GPUs: 同为扩散模型 (DiT) 的性能 / 配置优化 PR，涉及相同的 helios.py 模型文件和质量配置参数，属于同一功能线的维护。
- PR #37965 [diffusion] fix: preserve mapped courier tensor lifetime: 同为扩散模型的 bugfix PR，涉及 layerwise_offload 和张量生命周期管理，虽然具体问题不同，但都关注扩散模型的运行时稳定性和正确性。