PR #24028 完整报告

sgl-project/sglang

[NPU] [Diffusion] Use fused operator to improve Wan model E2E performance.

合并时间：2026-05-11 12:17

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24028

执行摘要

- 一句话：融合算子加速 NPU Wan 推理 2%-10%
- 推荐动作：该 PR 展示了如何为 NPU 后端添加融合算子路径，具有参考价值。但 review 中的重构建议未被采纳，对于追求高代码质量的团队尤其值得注意。整体改动较小，建议关注其中的模式设计。

功能与动机

根据 PR body，目的是使用 Triton 融合算子加速 Wan 系列模型的推理性能，具体包括 fused_scale_shift、fused_rsqrt_mul 和 fused_variance.

实现拆解

实现拆解

1. elementwise.py: 添加 ScaleShift 的 NPU 专用路径。在 ScaleShift 类中新增 forward_npu 方法，直接从 sgl_kernel_npu.norm.scale_shift 导入 fused_scale_shift 核函数，替代逐元素计算的 forward_native 路径。该核函数将乘加操作融合为单一算子。
2. layernorm.py: 为归一化层添加 NPU 专用路径。分别为 _ScaleResidualNormScaleShift 和 _NormScaleShift 类添加 forward_npu 方法。前者处理带残差连接的归一化，后者处理纯归一化 + 缩放平移。两者均使用 fused_scale_shift 核函数。
3. layernorm.py: 优化 tensor_parallel_rms_norm 函数。在 tensor_parallel_rms_norm 中，当 _is_npu 为真时，使用 fused_variance 替代 pow(2).mean，使用 fused_rsqrt_mul 替代 rsqrt 与乘法组合。这减少了内核启动和显存访问。
4. CI 脚本：更新 sgl-kernel-npu 版本。scripts/ci/npu/npu_ci_install_dependency.sh 中将版本标签从 2026.03.10.rc1 更新为 2026.05.01，并修正了下载路径使其包含 $PYTORCH_VERSION 变量，以确保正确的预编译包被安装。

这些改动均针对 NPU 后端，不影响其他硬件平台的逻辑。未新增测试文件，依赖 NPU CI 验证。

关键文件：

- python/sglang/multimodal_gen/runtime/layers/elementwise.py（模块 WAN 推理层；类别 source；类型 core-logic；符号 forward_npu）：新增 forward_npu 方法，实现 ScaleShift 在 NPU 上的融合算子调用，是性能优化的核心之一。

- python/sglang/multimodal_gen/runtime/layers/layernorm.py（模块 WAN 推理层；类别 source；类型 dependency-wiring；符号 forward_npu, tensor_parallel_rms_norm）：为主要归一化类添加 NPU 路径，并优化 tensor_parallel_rms_norm 使用融合核函数。
- scripts/ci/npu/npu_ci_install_dependency.sh（模块 NPU CI；类别 infra；类型 infrastructure）：更新 sgl-kernel-npu 包版本以支持新融合算子，确保 CI 能够正确安装测试依赖。

关键符号：forward_npu, tensor_parallel_rms_norm

关键源码片段

python/sglang/multimodal_gen/runtime/layers/elementwise.py

新增 forward_npu 方法，实现 ScaleShift 在 NPU 上的融合算子调用，是性能优化的核心之一。

```
class ScaleShift(CustomOp):
    """
    Fused kernel: a * (k + b) + c
    """
    def __init__(self, prefix: str = ""):
        super().__init__()

    def forward_native(
        self, a: torch.Tensor, b: torch.Tensor, c: torch.Tensor, k: int = 0
    ) -> torch.Tensor:
        # a.shape: [batch_size, seq_len, inner_dim]
        if b.dim() == 4:
            # b.shape: [batch_size, num_frames, 1, inner_dim]
            num_frames = b.shape[1]
            frame_seqlen = a.shape[1] // num_frames
            return c + (
                a.unflatten(dim=1, sizes=(num_frames, frame_seqlen)) * (k + b)
            ).flatten(1, 2)
        else:
            # b.shape: [batch_size, 1, inner_dim]
            return c + a * (k + b)

    def forward_cuda(
        self, a: torch.Tensor, b: torch.Tensor, c: torch.Tensor, k: int = 0
    ):
        return fuse_scale_shift_kernel(a, b, c, scale_constant=k)

    def forward_xpu(
        self, a: torch.Tensor, b: torch.Tensor, c: torch.Tensor, k: int = 0
    ):
        return self.forward_native(a, b, c, k=k)

    def forward_npu(
        self, a: torch.Tensor, b: torch.Tensor, c: torch.Tensor, k: int = 0
```

```
    ):
        # NPU 专用路径：使用 sgl_kernel_npu 中的 fused_scale_shift 融合核
        from sgl_kernel_npu.norm.scale_shift import fused_scale_shift

        return fused_scale_shift(a, b, c, scale_constant=k)
```

python/sglang/multimodal_gen/runtime/layers/layernorm.py

为主要归一化类添加 NPU 路径，并优化 tensor_parallel_rms_norm 使用融合核函数。

```
def tensor_parallel_rms_norm(x: torch.Tensor, norm: "RMSNorm") -> torch.Tensor:
    src_dtype = x.dtype
    weight = norm.weight.tensor_split(tp_size)[tp_rank].float()
    x_fp32 = x.float()
    if _is_npu:
        # NPU 路径：使用融合算子计算方差和归一化，减少内核启动
        from sgl_kernel_npu.norm.rmsnorm_split import fused_rsqrt_mul, fused_variance

        variance = fused_variance(x_fp32) # 使用融合方差计算
    else:
        variance = x_fp32.pow(2).mean(dim=-1, keepdim=True) # 原生 PyTorch
    variance = get_tp_group().all_reduce( # all_reduce 在条件外部，但实际代码在条件内部（PR
    未采纳建议）
        variance, op=torch._C._distributed_c10d.ReduceOp.AVG
    )
    if _is_npu:
        output = fused_rsqrt_mul(x_fp32, variance, weight, norm.variance_epsilon)
    else:
        output = x_fp32 * torch.rsqrt(variance + norm.variance_epsilon) * weight
    return output.to(dtype=src_dtype)

# 另外为 _ScaleResidualNormScaleShift 和 _NormScaleShift 添加了 forward_npu 方法
class _ScaleResidualNormScaleShift(CustomOp):
    def forward_npu(
        self, residual, x, gate, shift, scale
    ):
        from sgl_kernel_npu.norm.scale_shift import fused_scale_shift
        # 残差连接与 gate 处理逻辑同 native
        if isinstance(gate, int):
            assert gate == 1
            residual_output = residual + x
        elif isinstance(gate, torch.Tensor):
            if gate.dim() == 4:
                num_frames = gate.shape[1]
                frame_seqlen = x.shape[1] // num_frames
                residual_output = residual + (
                    x.unflatten(dim=1, sizes=(num_frames, frame_seqlen)) * gate
                ).flatten(1, 2)
            else:
                residual_output = residual + x * gate
```

```
        else:
            raise ValueError(f"Gate type {type(gate)} not supported")
        normalized = self.norm(residual_output)
        # 使用 NPU 融合核替代 fuse_scale_shift_kernel
        modulated = fused_scale_shift(normalized, scale, shift)
        return modulated, residual_output
```

评论区精华

review 中 gemini-code-assist[bot] 建议在 tensor_parallel_rms_norm 中将 all_reduce 调用提取到条件分支外以减少重复，但 PR 作者未采纳该建议，最终版本保留了分支内的 duplicated all_reduce。

- tensor_parallel_rms_norm 中 all_reduce 提取建议 (design): PR 作者未采纳此建议，最终版本保留了分支内的 duplicated all_reduce。

风险与影响

- 风险：主要风险包括：1）新版本 sgl-kernel-npu 包可能出现兼容性问题或缺失某些算子；2）NPU 专用路径只在 NPU CI 下测试，缺少独立单元测试；3）tensor_parallel_rms_norm 中的条件分支若 _is_npu 变量不准确或核函数有 bug，可能导致静默错误；4）性能优化对于大 SP 规模的效果减弱，可能存在边际收益。
- 影响：影响范围限制在 NPU 后端且使用 Wan 系列模型的用户。性能提升 2%-10%，具体取决于并行度（TP/SP）。CI 脚本变更影响所有 NPU 流水线，确保新内核包被安装。对其他硬件平台和模型无影响。
- 风险标记：缺少测试覆盖，依赖新 NPU 内核包

关联脉络

- 暂无明显关联 PR