PR #22786 完整报告

sgl-project/sglang

[AMD][diffusion] Add FlyDSL fused normalization kernels for ROCm diffusion models optimization

合并时间：2026-06-08 17:42

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/22786

执行摘要

- 一句话：为 AMD ROCm 添加 FlyDSL 融合归一化核，优化扩散模型
- 推荐动作：建议精读此 PR，尤其是 fused_residual_norm.py 中 FlyDSL 核的实现和寄存器缓存优化技术，以及 layernorm.py 中多级 fallback 的设计模式，值得在跨平台多后端开发中参考。

功能与动机

为 AMD ROCm 扩散模型优化归一化层性能。现有 CUDA/CUTLASS 融合核在 ROCm 上不可用，需要原生 ROCm 融合核以利用硬件潜力。PR body 明确指出 "Add AMD FlyDSL fused normalization kernels optimized for ROCm diffusion pipelines"。

实现拆解

1. 核心核实现：在 python/sglang/jit_kernel/diffusion/flydsl/fused_residual_norm.py 中使用 FlyDSL DSL 编写了两个核函数：flydsl_fused_residual_norm_scale_shift（融合残差加、门控乘、归一化、scale•shift）和 flydsl_norm_scale_shift（融合归一化、scale•shift）。核函数利用寄存器缓存优化，Phase 1 计算得到的 f32 中间值保留在寄存器中，Phase 2 直接重用，避免重新读取 HBM。
2. 编译缓存与 Fallback：通过 _get_or_compile 缓存编译结果，通过 launch_fused_norm 处理参数和降级。当维度不满足 5120 对齐或 flydsl 包未导入时，优雅降级到 native PyTorch 实现。
3. 集成到现有层：在 python/sglang/multimodal_gen/runtime/layers/layernorm.py 中，修改 RMSNorm 和 FusedScaleResidualNormScaleShift 的 forward_hip 方法，添加环境变量 SGLANG_USE_ROCM_FLYDSL 控制开关、导入保护和维度对齐检查。
4. 单元测试：新增 python/sglang/jit_kernel/tests/diffusion/test_flydsl_fused_norm.py，包含两个参数化测试用例，覆盖多种 norm_type、batch 和 seq_len 组合，对比 FlyDSL 输出与 PyTorch 参考输出，使用宽松容忍度（atol=5e-2, rtol=5e-2）。测试注册为 AMD CI。

关键文件：

- python/sglang/jit_kernel/diffusion/flydsl/fused_residual_norm.py（模块 融合核；类别 source；类型 core-logic；符号 _v, _build_fused_norm_module, flydsl_fused_residual_norm_ss_kernel, launch_fused_norm）：核心实现，定义了

FlyDSL 融合归一化核和包装函数

- python/sglang/multimodal_gen/runtime/layers/layernorm.py（模块 扩散层；类别 source；类型 dependency-wiring；符号 forward_hip）：修改 forward_hip 以集成 FlyDSL 核，包含环境变量开关、导入保护和维度降级
- python/sglang/jit_kernel/tests/diffusion/test_flydsl_fused_norm.py（模块 测试；类别 test；类型 test-coverage；符号 _ref_rms_norm, _ref_fused_residual_norm_ss, _ref_norm_ss, test_fused_residual_norm_scale_shift）：单元测试，验证 FlyDSL 核的正确性

关键符号：_build_fused_norm_module, flydsl_fused_residual_norm_ss_kernel, launch_fused_norm, _get_or_compile, flydsl_fused_residual_norm_scale_shift, flydsl_norm_scale_shift, forward_hip (RMSNorm), forward_hip (FusedScaleResidualNormScaleShift)

关键源码片段

python/sglang/jit_kernel/diffusion/flydsl/fused_residual_norm.py

核心实现，定义了 FlyDSL 融合归一化核和包装函数

```
"""FlyDSL fused normalization kernels for AMD ROCm (gfx950)."""

WARP_SIZE = 64
_VEC = 8
_NUM_WAVES = 10
FLYDSL_NORM_MIN_ALIGNED_DIM = WARP_SIZE * _NUM_WAVES * _VEC # 5120


def _build_fused_norm_module(D: int, is_rms: bool, has_gate: bool, has_weight: bool):
    # 设置 tiling 参数，要求 D 是 5120 的倍数
    VEC = _VEC
    NUM_WAVES = _NUM_WAVES
    BLOCK = NUM_WAVES * WARP_SIZE # 640
    assert D % FLYDSL_NORM_MIN_ALIGNED_DIM == 0, f"D must be multiple of {FLYDSL_
    NORM_MIN_ALIGNED_DIM}"
    NUM_ITERS = D // (BLOCK * VEC)

    @flyc.kernel(known_block_size=[BLOCK, 1, 1])
    def flydsl_fused_residual_norm_ss_kernel(
        y_ptr: fx.Tensor, res_out_ptr: fx.Tensor, res_ptr: fx.Tensor,
        x_ptr: fx.Tensor, gate_ptr: fx.Tensor, weight_ptr: fx.Tensor,
        bias_ptr: fx.Tensor, scale_ptr: fx.Tensor, shift_ptr: fx.Tensor,
        total_rows: Int32, gate_stride: Int32, scale_stride: Int32, shift_stride: Int32,
    ):
        row = fx.block_idx.x
        tid = fx.thread_idx.x
        # 创建 buffer 资源
        y_rsrc = buffer_ops.create_buffer_resource(y_ptr, max_size=True)
        # ... 省略中间代码 ...
```

```
        # Phase 1: 计算 residual + gate * x，累积部分和 / 平方和，保留 f32 值在寄存器
        # Phase 2: 使用寄存器中的 f32 值计算 RMSNorm/LayerNorm 和 scale•shift，
        # 避免重新读取 HBM，节省约 20% 带宽
    return flydsl_fused_residual_norm_ss_kernel
```

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

修改 forward_hip 以集成 FlyDSL 核，包含环境变量开关、导入保护和维度降级

```
def forward_hip(self, residual, x, gate, shift, scale):
    # 环境变量开关，默认不使用 FlyDSL
    if not _use_rocm_flydsl:
        return self.forward_native(residual, x, gate, shift, scale)
    # 导入保护：若 flydsl 包未安装则降级
    try:
        from sglang.jit_kernel.diffusion.flydsl.fused_residual_norm import (
            FLYDSL_NORM_MIN_ALIGNED_DIM,
            flydsl_fused_residual_norm_scale_shift,
        )
    except ImportError:
        return self.forward_native(residual, x, gate, shift, scale)
    # 维度对齐检查：仅当 hidden_size 是 5120 的倍数时才使用 FlyDSL
    if x.shape[-1] % FLYDSL_NORM_MIN_ALIGNED_DIM != 0:
        return self.forward_native(residual, x, gate, shift, scale)
    # 调用 FlyDSL 融合核
    return flydsl_fused_residual_norm_scale_shift(
        residual.contiguous(),
        x.contiguous(),
        gate.contiguous() if isinstance(gate, torch.Tensor) else None,
        _ensure_contiguous(self.norm.weight),
        _ensure_contiguous(self.norm.bias),
        scale.contiguous(),
        shift.contiguous(),
        self.norm_type,
        self.eps,
    )
```

评论区精华

Review 中主要有以下讨论：

- Dockerfile 依赖（HaiShaw）：要求添加 rocm.Dockerfile 安装 FlyDSL。作者通过兼容性修复（适配 flydsl>=0.1.5）解决了问题，最终无需修改 Dockerfile。
- 性能优化建议（gemini-code-assist[bot]）：建议将 batch 循环移至 GPU 网格以减少 CPU 开销、移除 forward_hip 中的冗余 contiguous 调用和调试打印、避免在 gate 为 None 时重复分配 dummy 张量、以及放宽维度对齐断言。这些建议并未全部被采纳，但最终 HaiShaw 批准了 PR。
- Dockerfile 安装 FlyDSL 依赖 (infra): 无需修改 Dockerfile，兼容性已解决。
- batch 循环 CPU 开销 (performance): 未明确是否采纳，但 PR 最终被批准。

- 连续调用和调试打印清理 (style): 未明确采纳，最终代码中仍保留 contiguous 调用。

风险与影响

- 风险：
  1. 维度对齐限制：核要求 hidden_size 是 5120 的倍数，不满足时会 fallback 到 native 实现。若用户模型 hidden_size 较小或不是 5120 倍数，无法获得加速。
  2. 集成路径未在 CI 中测试：PR-CI 未设置 SGLANG_USE_ROCM_FLYDSL 环境变量，从 layernorm.py 到 FlyDSL 核的集成分支从未被执行，可能引入未发现的兼容性问题。
  3. 可选依赖 flydsl：环境需要安装 flydsl 包（>=0.1.5），增加了部署复杂度。 - 影响：用户影响：AMD ROCm 用户启用环境变量后，在扩散模型（如 Wan2.2 T2V）推理的 Denoising 阶段可获得约 3-10% 的加速（总时间约 1-3%）。未启用或非 AMD 平台无影响。系统影响：新增约 1k 行 Python 代码，引入可选依赖 flydsl。团队影响：为 AMD 扩散优化建立了基础设施，未来可以类似方式添加更多融合核。 - 风险标记：维度对齐限制，集成路径未测试，可选依赖 flydsl

关联脉络

- 暂无明显关联 PR