PR #23938 完整报告

sgl-project/sglang

Optimize large GroupNorm SiLU apply

合并时间：2026-05-02 20:54

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23938

执行摘要

- 一句话：GroupNorm SiLU 大形状优化，VAE 解码加速 18x
- 推荐动作：建议合并。该 PR 展示了 Triton kernel 优化的典型手法：通过分析访问模式简化地址计算和寄存器使用。新增的 benchmark 脚本便于未来回归和调优。值得关注的设计决策是使用条件分支选择不同 kernel 而非统一路径，以及将 num_warps 从 8 降低到 4 的权衡。

功能与动机

HunyuanVideo VAE decode 中 triton_group_norm_silu 的 apply 阶段在大形状下存在性能瓶颈：现有通用 apply kernel 每个元素计算 idx // spatial_size 进行按通道的 affine 加载，导致整数除法和向量加载开销。PR 旨在通过提供 scalar-affine 变体来消除这一开销，提升 VAE 解码速度。

实现拆解

1. 新增标量 affine 应用 kernel _group_norm_apply_scalar_affine_kernel（文件 python/sglang/jit_kernel/diffusion/triton/group_norm_silu.py）：该 kernel 假设每个 TRITON program 覆盖从 chunk_start 开始的连续元素，且 chunk_start 与 spatial_size 对齐，因此一个 program 内所有元素都属于同一个 channel。基于此，weight 和 bias 在循环外一次性加载，避免每元素 channel_offsets = weight_group_offset + idx // spatial_size 计算和向量 affine load。
2. 在 _launch_chunked 函数中添加路由逻辑：当 spatial_size % _CHUNK_SIZE == 0 且 chunks_per_row >= 64 时，优先使用新 kernel，否则 fallback 到原 _group_norm_apply_kernel。新 kernel 的 num_warps 从 8 降低至 4，num_stages 保持 3。
3. 新增独立 benchmark 脚本 python/sglang/jit_kernel/benchmark/diffusion/bench_group_norm_silu.py：定义了包括 hunyuan_video_large 在内的 5 个基准 case，提供 run_case 进行正确性验证和性能统计。脚本通过 register_cuda_ci 注册到 CI（但默认 disabled），可在需要时手动执行。
4. 性能验证：H200 上运行 benchmark，新 kernel 在 hunyuan_video_large case 上达到 18.156x 加速；NCU 数据显示 apply 阶段 DRAM 吞吐从 51.86% 提升至 74.13%。单元测试（14 passed）和 HunyuanVideo 生成质量验证均通过。

关键文件：

- python/sglang/jit_kernel/diffusion/triton/group_norm_silu.py（模块 JIT 核；类别 source；类型 core-logic；符号 _group_norm_apply_scalar_affine_kernel, _launch_chunked）：核心修改文件，新增标量 affine 应用 kernel 并添加条件路由，实现性能优化主体。
- python/sglang/jit_kernel/benchmark/diffusion/bench_group_norm_silu.py（模块 性能基准；类别 source；类型 benchmark；符号 Case, dtype_from_name, dtype_name, parse_dtypes）：新增 benchmark 脚本，提供多种 shape 的正确性与性能评测基础设施。

关键符号：_group_norm_apply_scalar_affine_kernel, _launch_chunked, run_case, make_inputs, native_group_norm_silu

关键源码片段

python/sglang/jit_kernel/diffusion/triton/group_norm_silu.py

核心修改文件，新增标量 affine 应用 kernel 并添加条件路由，实现性能优化主体。

```
# 文件：python/sglang/jit_kernel/diffusion/triton/group_norm_silu.py

@triton.jit
def _group_norm_apply_scalar_affine_kernel(
    input_ptr, weight_ptr, bias_ptr, output_ptr, stats_ptr,
    channels, spatial_size, num_groups, channels_per_group, group_size,
    chunks_per_row,
    BLOCK_SIZE: tl.constexpr,
    BLOCKS_PER_PROGRAM: tl.constexpr,
):
    """
    标量 affine 应用 kernel，适用于 spatial_size % CHUNK_SIZE == 0 的对齐大形状。
    每个 program 覆盖连续元素，且每个 program 内的所有元素属于同一个 channel，
    因此 weight 和 bias 在循环外一次性加载，避免每元素 idx // spatial_size 计算。
    """
    row = tl.program_id(0).to(tl.int64)
    chunk_id = tl.program_id(1).to(tl.int64)

    batch_id = row // num_groups
    group_id = row - batch_id * num_groups
    chunk_start = chunk_id * BLOCK_SIZE * BLOCKS_PER_PROGRAM
    group_base = batch_id * channels * spatial_size + group_id * group_size

    # 利用对齐假设确定 channel_id
    channel_id = chunk_start // spatial_size
    affine_offset = group_id * channels_per_group + channel_id
    weight = tl.load(weight_ptr + affine_offset).to(tl.float32)
    bias = tl.load(bias_ptr + affine_offset).to(tl.float32)

    mean = tl.load(stats_ptr + row * 2)
    rstd = tl.load(stats_ptr + row * 2 + 1)
```

```
    offsets = tl.arange(0, BLOCK_SIZE)

    for block_id in range(BLOCKS_PER_PROGRAM):
        idx = chunk_start + block_id * BLOCK_SIZE + offsets
        mask = idx < group_size
        x = tl.load(input_ptr + group_base + idx, mask=mask, other=0.0).to(tl.float32)
        y = (x - mean) * rstd
        y = y * weight + bias
        y = y * tl.sigmoid(y)
        tl.store(output_ptr + group_base + idx, y, mask=mask)


# 在 _launch_chunked 函数中的路由逻辑（位于相同文件）：
def _launch_chunked(x_flat, weight, bias, num_groups, eps):
    # ... 前面的 partial sum 和 stats kernel 调用 ...

    # 条件分支：对齐大形状使用 scalar-affine kernel，否则 fallback 到通用版本
    if spatial_size % _CHUNK_SIZE == 0 and chunks_per_row >= 64:
        _group_norm_apply_scalar_affine_kernel[(rows, chunks_per_row)](
            x_flat, weight, bias, y_flat, stats,
            channels, spatial_size, num_groups, channels_per_group, group_size,
            chunks_per_row,
            BLOCK_SIZE=_BLOCK_SIZE,
            BLOCKS_PER_PROGRAM=_BLOCKS_PER_PROGRAM,
            num_warps=4, # 相比旧 kernel 的 8，减少寄存器压力
            num_stages=3,
        )
    else:
        _group_norm_apply_kernel[(rows, chunks_per_row)](
            x_flat, weight, bias, y_flat, stats,
            channels, spatial_size, num_groups, channels_per_group, group_size,
            chunks_per_row,
            BLOCK_SIZE=_BLOCK_SIZE,
            BLOCKS_PER_PROGRAM=_BLOCKS_PER_PROGRAM,
            num_warps=8,
            num_stages=3,
        )
    return y_flat
```

python/sglang/jit_kernel/benchmark/diffusion/bench_group_norm_silu.py

新增 benchmark 脚本，提供多种 shape 的正确性与性能评测基础设施。

```
# 文件：python/sglang/jit_kernel/benchmark/diffusion/bench_group_norm_silu.py

@dataclass(frozen=True)
class Case:
    name: str
    shape: tuple[int, ...]
    num_groups: int
```

```
# 目标优化 case 在最后
CASES = [
    Case("token_2d", (4, 128), 32),
    Case("image_2d", (2, 64, 32, 32), 32),
    Case("video_3d_small", (1, 64, 4, 16, 16), 32),
    Case("threshold_3d", (1, 128, 1, 256, 256), 32),
    Case("hunyuan_video_large", (1, 128, 20, 256, 256), 32),
]

def tolerance(dtype):
    """根据数据类型返回 atol, rtol"""
    if dtype == torch.float32:
        return 1e-5, 1e-5
    if dtype == torch.bfloat16:
        return 7e-2, 2e-2
    return 3e-3, 3e-3

def native_group_norm_silu(x, weight, bias, num_groups):
    """PyTorch 原生 reference 实现"""
    return F.silu(F.group_norm(x, num_groups, weight=weight, bias=bias, eps=EPS))

def make_inputs(case, dtype):
    """可复现的随机输入生成"""
    generator = torch.Generator(device='cuda')
    seed = len(case.shape) * 1009 + case.shape[1] * 17 + case.num_groups
    generator.manual_seed(seed)
    x = torch.randn(case.shape, device='cuda', dtype=dtype, generator=generator)
    weight = torch.randn(case.shape[1], device='cuda', dtype=dtype, generator=generator)
    bias = torch.randn(case.shape[1], device='cuda', dtype=dtype, generator=generator)
    return x, weight, bias

def run_case(case, dtype, rounds, warmup, rep):
    """
    对单个 case 执行正确性验证和性能 benchmark。
    返回包含 median 耗时和加速比的字典。
    """
    x, weight, bias = make_inputs(case, dtype)
    with torch.inference_mode():
        # 正确性验证
        actual = triton_group_norm_silu(x, weight, bias, num_groups=case.num_groups, eps=EPS)
        expected = native_group_norm_silu(x, weight, bias, case.num_groups)
        atol, rtol = tolerance(dtype)
        torch.testing.assert_close(actual, expected, atol=atol, rtol=rtol)
        # 性能 benchmark（多轮取中位数）
        native_stats = []
        fused_stats = []
        for _ in range(rounds):
            native_stats.append(do_bench_us(
                lambda: native_group_norm_silu(x, weight, bias, case.num_groups),
```

```
            warmup=warmup, rep=rep))
        fused_stats.append(do_bench_us(
            lambda: triton_group_norm_silu(x, weight, bias, num_groups=case.num_groups, eps=
            EPS),
            warmup=warmup, rep=rep))
    native_median = summarize([s[0] for s in native_stats])
    fused_median = summarize([s[0] for s in fused_stats])
    return {
        "case": case.name,
        "dtype": dtype_name(dtype),
        "native_us": native_median,
        "fused_us": fused_median,
        "speedup": native_median / fused_median,
    }
```

评论区精华

PR 无人工 review 讨论，仅 gemini-code-assist[bot] 自动审查并声明无反馈。作者 BBuf 在 issue 评论中触发 CI retag 和执行。无争议点。

- 自动代码审查 (other): 无人工反馈，无需执行操作。

风险与影响

- 风险： 1) 功能风险：新 kernel 仅在 spatial_size % _CHUNK_SIZE == 0 and chunks_per_row >= 64 时启用，其他情况 fallback 到旧 kernel，不会产生错误。但条件判断可能因 future 配置变化导致路径错误； 2) 性能风险：新 kernel 优化针对特定形状，非对齐形状使用旧 kernel，性能无退化； 3) 测试覆盖：benchmark 包含多种 case，但单元测试仅覆盖 H200 GPU，Triton 不同版本或 GPU 架构可能暴露差异； 4) 兼容性：kernel 依赖 Triton 编译器，Triton 升级可能导致 kernel 行为变化。
  - 影响：用户影响：使用 HunyuanVideo 进行 VAE 解码的用户获得约 18x 的 apply 阶段加速，整体视频生成延迟降低。其他 diffusion 模型不受影响。系统影响：新增 benchmark 文件仅用于开发 /CI，不影响运行时。团队影响：基于条件的 kernel 选择模式可在未来优化中复用。
  - 风险标记：条件分支依赖，Triton 版本兼容，测试覆盖局限

关联脉络

- PR #23148 diffusion: enable group norm silu fuse by default: 同一个 GroupNorm SiLU 融合核的先前 PR，当前 PR 在此基础上进一步优化 apply 阶段。