PR #36521 完整报告

sgl-project/sglang

[diffusion][kernel] avoid 4D scale-shift autotuning

合并时间：2026-08-28 16:58

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/36521

执行摘要

- 一句话：4D scale-shift 去请求时 autotuning，冷启动提速 21%
- 推荐动作：值得精读的 kernel 性能优化案例。核心价值在于"对带宽受限、每步调用的内核，请求期 autotune 成本超过收益"这一判断方法，以及配套的 ABBA 冷启动对比、输出哈希一致性、CI benchmark 三位一体的验证方式。建议后续在同类小内核（如 elementwise、逐 token 归一化）中复用该模式，并保持"小内核静态化 + 大内核 autotune"的分类原则。

功能与动机

PR body 明确指出：LingBot-World 每个 transformer block 都会调用这个带宽受限内核，完整 H100 trace 中 199 次 Python 调用扩展为 3820 次 Triton autotune 启动，tuning 主导了第一个 denoise 步骤；而选定的 64/128/256/512 tile 在所有测试生产形状上逐位一致，封顶 2 的幂选择最快或在测量噪声内，因此可以安全地静态化。

实现拆解

1. 变更入口：核心修改在 python/sglang/kernels/ops/diffusion/modulate/scale_shift_triton.py 的 fuse_scale_shift_kernel 与其 4D 内核 _fused_scale_shift_4d_kernel，该路径服务于 Causal Wan、LingBot-World 等 causal video 模型的逐帧 modulation。
2. 移除请求时 autotuning：删除 _fused_scale_shift_4d_kernel 上的 @triton.autotune 装饰器及其 5 组 BLOCK_N/num_warps 配置；在宿主侧用 block_n = max(64, min(512, triton.next_power_of_2(C))) 计算静态 tile，num_warps 按 block_n == 64 选 2、其余选 4，网格改为 (rows, cdiv(C, block_n)) 并通过内核参数显式传入 BLOCK_N 与 num_warps。这样每次请求不再触发多轮编译，冷启动成本被消除。
3. 新增 4D 正确性测试：test/registered/kernels/ops/diffusion/test_modulate.py 增加 SCALE_SHIFT_4D_CASES（覆盖小维度、多 batch、Causal Wan 与 LingBot-World 生产形状）和 test_scale_shift_4d_matches_torch，参数化 bf16/fp16 与 scale_constant 0/1，与 torch reference（unflatten 后乘加再 flatten）用 atol=5e-2, rtol=5e-2 对比，守护 close 契约。
4. 新增生产形状 benchmark：新增 test/registered/kernels/benchmark/diffusion/bench_scale_shift_4d.py，定义 FULL_WORKLOADS（wan、sana_video、longlive、lingbot_world）与 CI_WORKLOADS，用 cuda_event_us 取多轮中位数对比 torch 与 triton，并输出 cold ms 与 speedup；通过 register_cuda_ci 注册为 base-b-kernel-benchmark，CI 只跑缩减集。

5. 文档配套：python/sglang/kernels/ops/diffusion/README.md 与 python/sglang/multimodal_gen/.claude/skills/sglang-diffusion-benchmark-profile/existing-fast-paths.md 记录“4D 路径使用静态封顶 2 的幂 tile、不要 reintroduce 请求时 autotuning”的约束，防止后续重构回退。

关键文件：

- python/sglang/kernels/ops/diffusion/modulate/scale_shift_triton.py（模块 扩散算子；类别 source；类型 core-logic；符号 fuse_scale_shift_kernel, _fused_scale_shift_4d_kernel）：核心源码变更：移除 @triton.autotune，改为宿主侧封顶 2 的幂 tile 启发式，直接决定冷启动性能收益。
- test/registered/kernels/benchmark/diffusion/bench_scale_shift_4d.py（模块 性能基准；类别 test；类型 benchmark；符号 Workload, cuda_event_us, benchmark）：新增生产形状 benchmark 并注册进 CI，为去 autotune 的性能声明提供可复现证据。
- test/registered/kernels/ops/diffusion/test_modulate.py（模块 内核测试；类别 test；类型 test-coverage；符号 test_scale_shift_4d_matches_torch）：新增 4D bf16/fp16 正确性测试，覆盖生产形状与 scale_constant 两种取值，守护 close 契约。
- python/sglang/kernels/ops/diffusion/README.md（模块 内核文档；类别 docs；类型 documentation）：记录 fuse_scale_shift_kernel 的静态 tile 契约，防止未来 reintroduce autotuning。
- python/sglang/multimodal_gen/.claude/skills/sglang-diffusion-benchmark-profile/existing-fast-paths.md（模块 技能文档；类别 docs；类型 documentation）：在 benchmark/profile skill 文档中固化冷启动约束，告知后续开发者不要为 4D 路径恢复请求时 autotuning。

关键符号：fuse_scale_shift_kernel, _fused_scale_shift_4d_kernel, test_scale_shift_4d_matches_torch, benchmark, cuda_event_us

关键源码片段

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

核心源码变更：移除 @triton.autotune，改为宿主侧封顶 2 的幂 tile 启发式，直接决定冷启动性能收益。

```
# python/sglang/kernels/ops/diffusion/modulate/scale_shift_triton.py
# 4D causal-video scale/shift：x 形状 [B, L, C]，scale 形状 [B, F, 1, C] 按帧广播，
# shift 与 x 同形（逐 token）。LingBot-World 每个 transformer block 都会调用它。

# 改造前 _fused_scale_shift_4d_kernel 挂着 @triton.autotune，在请求期为每个新 shape
# 扫描 5 组 BLOCK_N（64~1024）并各自编译，冷启动成本远超带宽受限内核本身；
# 因此移除 autotune，改由宿主侧静态导出 tile。内核内部计算逻辑保持不变。

@triton.jit
def _fused_scale_shift_4d_kernel(
    output_ptr,
    x_ptr,
    scale_ptr,
```

```
    shift_ptr,
    num_frames: tl.constexpr,
    frame_seqlen: tl.constexpr,
    scale_constant: tl.constexpr,
    rows: tl.constexpr,
    C: tl.constexpr,
    BLOCK_N: tl.constexpr, # 由宿主导出，不再参与 autotune 搜索
):
    # 按（行，列块）二维网格加载 x/scale/shift，计算
    # x * (scale_constant + scale) + shift 后写回，实现细节见原文件。
    ...


def fuse_scale_shift_kernel(x, scale, shift, scale_constant=1.0):
    # ... 前置合法性检查与其他维度分支省略，此处只展示 4D 分支的 tile 决策 ...

    rows = ...
    C = ...
    # 封顶 2 的幂 tile：生产 hidden 尺寸（1536/2240/3072/5120）下
    # 与 autotune 选出的 64/128/256/512 结果位一致，且最快或在测量噪声内。
    block_n = max(64, min(512, triton.next_power_of_2(C)))
    num_warps = 2 if block_n == 64 else 4 # 小 tile 减少 warp 数避免资源浪费
    grid = (rows, triton.cdiv(C, block_n))

    num_frames = scale.shape[1]
    assert L % num_frames == 0
    frame_seqlen = L // (rows * num_frames)
    _fused_scale_shift_4d_kernel[grid](
        output,
        x,
        scale,
        shift,
        num_frames=num_frames,
        frame_seqlen=frame_seqlen,
        scale_constant=scale_constant,
        rows=rows,
        C=C,
        BLOCK_N=block_n,
        num_warps=num_warps,
    )
    return output
```

test/registered/kernels/benchmark/diffusion/bench_scale_shift_4d.py

新增生产形状 benchmark 并注册进 CI，为去 autotune 的性能声明提供可复现证据。

```
# test/registered/kernels/benchmark/diffusion/bench_scale_shift_4d.py
# 生产形状与 CI 形状分开：CI 只跑缩减集以控制耗时，本地跑完整集以获得可信数据。

@dataclass(frozen=True)
```

```
class Workload:
    name: str
    shape: tuple[int, int, int]
    num_frames: int


FULL_WORKLOADS = [
    Workload("wan_s24960_c1536", (1, 24960, 1536), 5),
    Workload("sana_video_s7800_c2240", (1, 7800, 2240), 5),
    Workload("longlive_s1560_c3072", (1, 1560, 3072), 3),
    Workload("lingbot_world_s4680_c5120", (1, 4680, 5120), 1),
]
CI_WORKLOADS = [
    Workload("ci_s1024_c1536", (1, 1024, 1536), 4),
    Workload("ci_s512_c5120", (1, 512, 5120), 2),
]


def cuda_event_us(fn, warmups: int, repeats: int, rounds: int) -> float:
    # 预热后使用 CUDA event 计时，多轮取中位数以抵抗系统噪声。
    for _ in range(warmups):
        fn()
    torch.cuda.synchronize()

    samples = []
    for _ in range(rounds):
        start = torch.cuda.Event(enable_timing=True)
        end = torch.cuda.Event(enable_timing=True)
        start.record()
        for _ in range(repeats):
            fn()
        end.record()
        end.synchronize()
        samples.append(start.elapsed_time(end) * 1000.0 / repeats)
    samples.sort()
    return samples[len(samples) // 2]
```

test/registered/kernels/ops/diffusion/test_modulate.py

新增 4D bf16/fp16 正确性测试，覆盖生产形状与 scale_constant 两种取值，守护 close 契约。

```
# test/registered/kernels/ops/diffusion/test_modulate.py
# Causal Wan 与 LingBot 使用逐帧 4D modulation + 逐 token shift,
# 需要与 torch reference 对齐的 close 契约测试（非 bit-exact）。

SCALE_SHIFT_4D_CASES = [
    ((1, 18, 96), 3), # 小维度覆盖边界
    ((2, 20, 384), 4), # batch > 1，多帧
    ((1, 9, 1536), 3), # Causal Wan 形状
    ((1, 4, 5120), 2), # LingBot-World 生产形状
```

```
]


@pytest.mark.parametrize("shape,num_frames", SCALE_SHIFT_4D_CASES)
@pytest.mark.parametrize("dtype", [torch.bfloat16, torch.float16])
@pytest.mark.parametrize("scale_constant", [0, 1])
def test_scale_shift_4d_matches_torch(shape, num_frames, dtype, scale_constant):
    batch, seq_len, hidden = shape
    x = torch.randn(shape, device=DEVICE, dtype=dtype)
    scale = torch.randn((batch, num_frames, 1, hidden), device=DEVICE, dtype=dtype)
    shift = torch.randn_like(x)

    # reference：按帧 unflatten 后做 x * (scale_constant + scale) + shift，再 flatten。
    frame_seqlen = seq_len // num_frames
    expected = (
        x.unflatten(1, (num_frames, frame_seqlen)) * (scale_constant + scale)
        + shift.unflatten(1, (num_frames, frame_seqlen))
    ).flatten(1, 2)
    actual = fuse_scale_shift_kernel(x, scale, shift, scale_constant)

    # close 契约：tile 选择不影响数值，但内核不保证逐位一致。
    torch.testing.assert_close(actual, expected, atol=5e-2, rtol=5e-2)
```

评论区精华

该 PR 没有任何 review 评论线程（comments_count 为 0、review_comments_count 为 0），由作者 BBuf 直接合入。设计决策的证据全部沉淀在 PR body 与代码注释中：ABBA 对比使用独立 Triton cache 让 baseline 与 candidate 都承担真实冷启动成本；四个 MP4 输出 SHA256 一致，验证无数值漂移；并明确说明 LingBot causal DMD 不支持 breakable CUDA graphs，因此不做 BCG 性能声明。

- 暂无高价值评论线程

风险与影响

- 风险：
  - 静态 tile 对未知形状可能非最优：block_n 被限制在 64~512，对 C 大于 512 的宽 hidden（如 5120）固定为 512，未在更多形状上验证最优性；PR 声明所测生产形状下最快或在噪声内，但未来新增 hidden 尺寸可能变慢，需依赖 CI benchmark 监控。
  - 数值契约是 close 而非 bit-exact：4D 测试使用 torch.testing.assert_close(atol=5e-2) 而非 torch.equal，若未来模型对逐位一致性敏感需要额外关注。
  - 硬件泛化不足：性能数据仅来自 H100；test_modulate.py 同时注册了 AMD CI，AMD 上该 tile 启发式的表现未单独验证。
  - 回归风险：移除 autotune 后失去 per-shape 自适应能力，冷启动收益依赖 Triton cache 行为，cache 未命中时的改善在 benchmark 中已有量化（1338 ms 到 715 ms），但其他环境需要复测。

- 影响：
  - 用户与模型：LingBot-World、Causal Wan、SANA Video、LongLive 等 causal video 模型首个 denoise 步骤显著加速，H100 上请求 E2E 提升约 7.64%，首帧延迟改善明显。
  - 系统：每请求数千次 autotune 启动消失，降低 Triton 编译缓存压力与 CPU 侧调度开销，对多请求并发场景有利。
  - 团队：新增的 CI benchmark（base-b-kernel-benchmark）与单元测试为内核数值回归和 tile 选择退化提供持续监控；文档明确约束，降低后续重构引入 autotune 的风险。
  - 风险标记：静态 tile 对未知形状可能非最优，性能数据仅基于 H100，4D 路径为 close 契约非 bit-exact，AMD 路径未单独验证

关联脉络

- PR #36726 [Diffusion] Fix the five unit tests failing on main: 同属 diffusion 内核与运行时质量线，修复了多个 diffusion 单元测试，与本 PR 的 4D kernel 测试配套共同稳定 diffusion CI。
- PR #36658 [multimodal_gen] fix: make tail_attn_meta CUDA-graph capturable: 同为 diffusion/multimodal_gen 性能与冷启动相关修复，关注首步开销与 CUDA graph 行为，与本 PR 的去 autotune 优化方向一致。
- PR #34747 [Cosmos3] Add cosmos3 transfer capability: diffusion pipeline 功能演进代表，后续新增模型与 benchmark 都可复用本 PR 建立的静态 tile 与基准测试模式。