PR #29390 完整报告

sgl-project/sglang

[Diffusion] Fuse LTX2 Ada values

合并时间：2026-06-26 23:13

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/29390

执行摘要

- 一句话：融合 LTX-2.3 Ada value 计算，端到端加速约 10%
- 推荐动作：值得精读。Triton 内核的编写方式（一次合并 9 个输出）和 fallback 模式（全局禁用标志 + 一次性异常抑制）是好的实践，可推广到类似计算模式。

功能与动机

LTX2TransformerBlock 中多次调用 get_ada_values 对 scale_shift_table 和 timestep 进行相同的切片加操作，产生冗余计算。融合后可减少重复的内存读写和计算，提升推理性能。该优化灵感来自 NVlabs/Sana sol-engine 的 LTX2 Ada-value fusion（PR body）。

实现拆解

1. 新增 Triton 内核 _ltx2_ada_values9_kernel：对每行、每 hidden 维度，同时加载 9 组 (scale_shift_table, timestep) 并相加，一次写入 9 个输出张量，避免重复计算和广播。
2. 添加包装函数 ltx2_ada_values9：校验输入合法性（维度、数据类型、连续性等），调用内核并返回 9 个 Ada 张量元组。
3. 在模型代码 ltx_2.py 中引入运行时禁用标志 _LTX2_FUSED_ADA_VALUES_RUNTIME_DISABLED 和守卫函数 _ltx2_try_fused_ada_values9：先检查快速路径条件（TP=1、CUDA、bf16、连续等），若满足则调用 Triton 内核；若执行失败则记录警告并永久禁用快速路径，回退到原始 PyTorch 实现。
4. 在 LTX2TransformerBlock.forward 中，对 video 和 audio 两个流分别尝试快速路径，若成功则使用返回的 Ada 张量元组切片代替原 get_ada_values 切片调用。
5. 新增测试文件 test_ltx2_ada_values.py：包含参数化正确性测试（bf16/fp32 表、多种 batch/seq/hidden）和形状拒绝测试，并通过 CUDA CI 注册。

关键文件：

- python/sglang/jit_kernel/diffusion/triton/ltx2_ada_values.py（模块 JIT 内核；类别 source；类型 dependency-wiring；符号 _ltx2_ada_values9_kernel, ltx2_ada_values9）：新增 Triton 内核，是性能优化的核心
- python/sglang/multimodal_gen/runtime/models/dits/ltx_2.py（模块 扩散模型；类别 source；类型 data-contract；符号 _ltx2_disable_fused_ada_values, _ltx2_try_fused_ada_values9）：集成快速路径和 fallback 机制，是性能优化生效的入口

- test/registered/jit/diffusion/test_ltx2_ada_values.py（模块 测试覆盖；类别 test；类型 test-coverage；符号 cuda_setup, _reference, test_ltx2_ada_values9, test_ltx2_ada_values9_rejects_unsupported_shape）：提供正确性测试和输入校验测试，保障优化质量

关键符号：_ltx2_ada_values9_kernel, ltx2_ada_values9, _ltx2_disable_fused_ada_values, _ltx2_try_fused_ada_values9, test_ltx2_ada_values9, _reference

关键源码片段

python/sglang/jit_kernel/diffusion/triton/ltx2_ada_values.py

新增 Triton 内核，是性能优化的核心

```
# _ltx2_ada_values9_kernel: 一次加载 9 组 table/temb 并相加，输出 9 个 Ada 张量
@triton.jit
def _ltx2_ada_values9_kernel(
    temb_ptr, table_ptr,
    out0_ptr, out1_ptr, out2_ptr, out3_ptr, out4_ptr,
    out5_ptr, out6_ptr, out7_ptr, out8_ptr,
    rows: tl.constexpr, hidden: tl.constexpr,
    total_params: tl.constexpr,
    table_stride_p: tl.constexpr, table_stride_d: tl.constexpr,
    BLOCK_N: tl.constexpr,
):
    row = tl.program_id(0).to(tl.int64)
    cols = tl.arange(0, BLOCK_N)
    mask = cols < hidden
    # 第 0 组
    table0 = tl.load(table_ptr + 0 * table_stride_p + cols * table_stride_d, mask=mask, other=0.0).
    to(tl.bfloat16)
    temb0 = tl.load(temb_ptr + row * total_params * hidden + 0 * hidden + cols, mask=mask,
    other=0.0).to(tl.bfloat16)
    # 第 1 组
    table1 = tl.load(table_ptr + 1 * table_stride_p + cols * table_stride_d, mask=mask, other=0.0).
    to(tl.bfloat16)
    temb1 = tl.load(temb_ptr + row * total_params * hidden + 1 * hidden + cols, mask=mask,
    other=0.0).to(tl.bfloat16)
    # ... 第 2-8 组，模式完全相同
    # 存储结果
    tl.store(out0_ptr + row * hidden + cols, (table0 + temb0).to(tl.bfloat16), mask=mask)
    tl.store(out1_ptr + row * hidden + cols, (table1 + temb1).to(tl.bfloat16), mask=mask)
    # ... 存储 out2-8

def ltx2_ada_values9(scale_shift_table: torch.Tensor, timestep: torch.Tensor) -> tuple[torch.
Tensor, ...]:
    # 输入校验：timestep 必须是 [B, S, 9*D] CUDA bf16 连续张量
    if timestep.ndim != 3:
        raise ValueError("timestep must have shape [B, S, 9 * D]")
    if not timestep.is_cuda or timestep.dtype != torch.bfloat16:
```

```
        raise ValueError("timestep must be a CUDA bfloat16 tensor")
    if not timestep.is_contiguous():
        raise ValueError("timestep must be contiguous")
    if scale_shift_table.ndim != 2 or scale_shift_table.shape[0] != 9:
        raise ValueError("scale_shift_table must have shape [9, D]")
    # ... 更多校验后，配置内核参数，启动内核
    outputs = [torch.empty(batch, seq, hidden, device='cuda', dtype=torch.bfloat16) for _ in
    range(9)]
    _ltx2_ada_values9_kernel[(batch * seq,)](timestep, scale_shift_table, *outputs, ...)
    return tuple(outputs)
```

python/sglang/multimodal_gen/runtime/models/dits/ltx_2.py

集成快速路径和 fallback 机制，是性能优化生效的入口

```
# 全局禁用标志，首次异常后永久关闭快速路径
_LTX2_FUSED_ADA_VALUES_RUNTIME_DISABLED = False

def _ltx2_disable_fused_ada_values(exc: Exception) -> None:
    global _LTX2_FUSED_ADA_VALUES_RUNTIME_DISABLED
    _LTX2_FUSED_ADA_VALUES_RUNTIME_DISABLED = True
    logger.warning_once(f"Disabling LTX2 fused Ada values fast path: {exc}")

def _ltx2_try_fused_ada_values9(scale_shift_table, batch_size, timestep):
    # 快速路径条件检查：仅当 TP=1、CUDA、bf16、连续、形状匹配时启用
    if (_LTX2_FUSED_ADA_VALUES_RUNTIME_DISABLED or get_tp_world_size() != 1
        or not timestep.is_cuda or timestep.dtype != torch.bfloat16
        or timestep.ndim != 3 or int(timestep.shape[0]) != int(batch_size)
        or not timestep.is_contiguous() or not scale_shift_table.is_cuda
        or scale_shift_table.dtype not in (torch.bfloat16, torch.float32)
        or scale_shift_table.ndim != 2 or int(scale_shift_table.shape[0]) != 9
        or scale_shift_table.stride(-1) != 1):
        return None
    hidden = int(scale_shift_table.shape[1])
    if hidden % 256 != 0 or hidden > 8192 or timestep.shape[-1] != 9 * hidden:
        return None
    try:
        from sglang.jit_kernel.diffusion.triton.ltx2_ada_values import ltx2_ada_values9
        return ltx2_ada_values9(scale_shift_table, timestep)
    except Exception as exc:
        _ltx2_disable_fused_ada_values(exc)
        return None

# 在 forward 中使用（video 流为例）：
video_ada_values = _ltx2_try_fused_ada_values9(self.scale_shift_table, batch_size, temb)
if video_ada_values is None:
    vshift_msa, vscale_msa, vgate_msa = self.get_ada_values(self.scale_shift_table, batch_size,
    temb, slice(0, 3))
else:
    vshift_msa, vscale_msa, vgate_msa = video_ada_values[0:3] # 直接取前 3 个
```

评论区精华

Reviewer mickqian 表示 "impressive. we should utilize AI more to find these redundant computation"，充分肯定优化价值。无其他讨论。

- 暂无高价值评论线程

风险与影响

- 风险：Triton 内核依赖 CUDA 和特定硬件，在非 CUDA 平台（如 CPU、NPU）上不可用，但 fallback 机制确保静默回归原始路径。核心推理路径 ltx_2.py 的 forward 逻辑被修改，但测试覆盖 +fallback 降低了风险。新内核的数值精度经过 SSIM/PSNR 验证与原始路径等价。
- 影响：对 LTX-2.3 模型用户，默认获得约 10% 的端到端加速，无需任何配置。影响范围局限在 diffusion 模型的 LTX2 系列。系统性能无负面影响。
- 风险标记：Triton 内核依赖，硬件兼容性，核心路径变更

关联脉络

- PR #27420 Diffusion: add JoyEcho multi-shot A/V generation support: 同一文件 ltx_2.py 的变更基础，引入 LTX2 模型支持，本 PR 在其上做性能优化