PR #23248 完整报告

sgl-project/sglang

[NPU][diffusion] add selectable parallel VAE decode strategies

合并时间：2026-05-08 02:37

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23248

执行摘要

- 一句话：为 Qwen-Image VAE 解码添加可选择的并行策略
- 推荐动作：值得精读，特别是 fused 自定义算子的设计模式（CUDA Triton + PyTorch fallback）和并行策略选择逻辑。对于希望扩展 VAE 解码到其他模型的开发者有参考价值。

功能与动机

PR body 指出："This PR follows up on the community review suggestions from the previous Qwen-Image parallel decoding work. ... The goal is to keep the default behavior unchanged while allowing Qwen-Image to choose the most suitable parallel decode path for different image sizes."

实现拆解

1. 配置扩展：在 VAEConfig（base.py）中新增 use_parallel_decode（默认 False）和 parallel_decode_mode（可选 tiled / patch / auto），并注册 CLI 参数 --vae-config.use-parallel-decode 和 --vae-config.parallel-decode-mode。
2. 基类重构：在 ParallelTiledVAE（common.py）中添加配置属性，新增 parallel_patch_decode 方法（将 latent 切分为 patch 并分配到各 GPU 解码），将原有的数据收集 / 合并逻辑（_parallel_data_generator、_merge_parallel_tiled_results）替换为更通用的 _process_parallel_tiled_outputs，并修正 latent 宽度计算 bug。
3. Qwen-Image VAE 解码调度：在 autoencoder_kl_qwenimage.py 中重写 _decode_with_parallel_dispatch，根据 parallel_decode_mode（auto 时根据图像尺寸自动选择）调用 parallel_patch_decode 或 parallel_tiled_decode，同时移除原先在子类中的 _process_parallel_tiled_outputs（已上移至基类）。
4. 新增 fused 自定义算子：新建 fused_scale_shift_gate.py，注册 FusedLayerNormScaleShiftGateSelect01 和 FusedResidualLayerNormScaleShiftGateSelect01，CUDA 路径调用 Triton kernel，HIP 及其他平台使用 PyTorch fallback。
5. Qwen-Image DiT 模型适配：在 qwen_image.py 中移除对 sglang.jit_kernel.diffusion.triton.scale_shift 的直接导入和 current_platform.is_hip() 的平台判断，改用新的 fused 算子实例，在 _modulate 方法中通过 is_scale_residual 分支统一调用。

关键文件：

- python/sglang/multimodal_gen/runtime/models/vaes/common.py（模块 VAE 基础；类别 source；类型 data-contract；符号 _parallel_data_generator, _merge_parallel_tiled_results, _process_parallel_tiled_outputs, parallel_patch_decode）：核心基类 ParallelTiledVAE 重构：新增 parallel_patch_decode 方法，将并行解码上层逻辑集中到基类，移除旧的私有辅助函数
- python/sglang/multimodal_gen/runtime/layers/fused_scale_shift_gate.py（模块 融合算子；类别 source；类型 core-logic；符号 FusedLayerNormScaleShiftGateSelect01, forward_cuda, forward_hip, forward_native）：新增的自定义算子模块，封装 Fusion LayerNorm + Scale/Shift + Gate 操作，提供 CUDA Triton 和通用 fallback
- python/sglang/multimodal_gen/runtime/models/vaes/autoencoder_kl_qwenimage.py（模块 QwenVAE；类别 source；类型 data-contract；符号 _process_parallel_tiled_outputs）：重写 _decode_with_parallel_dispatch，根据模式选择调用基类的 patch 或 tiled 解码；移除原本在子类中的 _process_parallel_tiled_outputs
- python/sglang/multimodal_gen/runtime/models/dits/qwen_image.py（模块 DiT 模型；类别 source；类型 data-contract）：改用新 fused 算子替代直接 Triton kernel 调用，简化 _modulate 方法，移除平台判断
- python/sglang/multimodal_gen/configs/models/vaes/base.py（模块 VAE 配置；类别 source；类型 data-contract）：VAE 并行解码配置入口，新增 use_parallel_decode 和 parallel_decode_mode 字段及 CLI 参数

关键符号：ParallelTiledVAE.parallel_patch_decode, ParallelTiledVAE.parallel_tiled_decode, ParallelTiledVAE._process_parallel_tiled_outputs, QwenImageVAE._decode_with_parallel_dispatch, FusedLayerNormScaleShiftGateSelect01.forward_cuda, FusedLayerNormScaleShiftGateSelect01.forward_native, FusedResidualLayerNormScaleShiftGateSelect01.forward_cuda, FusedResidualLayerNormScaleShiftGateSelect01.forward_native, QwenImageJointTransformerBlock._modulate

关键源码片段

python/sglang/multimodal_gen/runtime/models/vaes/common.py

核心基类 ParallelTiledVAE 重构：新增 parallel_patch_decode 方法，将并行解码上层逻辑集中到基类，移除旧的私有辅助函数

```
# python/sglang/multimodal_gen/runtime/models/vaes/common.py
# (head 版本关键片段 )
class ParallelTiledVAE(ABC, nn.Module):
    # 新增配置属性
    use_parallel_decode: bool
    parallel_decode_mode: str

    def __init__(self, config: VAEConfig, **kwargs) -> None:
        super().__init__()
        self.config = config
```

```
    # ... 原有属性 ...
    self.use_parallel_decode = config.use_parallel_decode
    self.parallel_decode_mode = config.parallel_decode_mode

def parallel_tiled_decode(self, z: torch.FloatTensor) -> torch.FloatTensor:
    """
    Parallel version of tiled_decode that distributes both temporal
    and spatial computation across GPUs.

    此方法已被重构为调用通用的 _process_parallel_tiled_outputs,
    并修正了 latent 宽度索引错误（使用 z.shape[-1] 替代错误的变量）。
    """
    world_size, rank = get_sp_world_size(), get_sp_parallel_rank()
    _, _, T, H, W = z.shape
    # ... 计算 tile 划分 ...
    # 每个 rank 处理部分 tile，收集后合并
    local_results = []
    local_dim_metadata = []
    # ... 分配本地 tile 并调用 self._decode ...
    # 改为调用基类通用合并方法
    return self._process_parallel_tiled_outputs(...)

def parallel_patch_decode(self, z: torch.FloatTensor) -> torch.FloatTensor:
    """
    Patch parallel decode: split latent into patches along H/W,
    each GPU decodes its patch independently, then gather & merge.

    对于小图，patch 模式比 tiled 模式更高效（减少 tile 重叠计算）。
    """
    world_size, rank = get_sp_world_size(), get_sp_parallel_rank()
    _, _, T, H, W = z.shape
    # 将 H,W 切成 world_size 个 patch（使用 isqrt 尽量保持方形）
    num_patches = world_size
    patch_rows = isqrt(num_patches)
    while num_patches % patch_rows != 0:
        patch_rows -= 1
    patch_cols = num_patches // patch_rows

    p_h = H // patch_rows
    p_w = W // patch_cols
    # 每个 rank 解码自己负责的 patch
    local_patches = []
    for i in range(patch_rows):
        for j in range(patch_cols):
            if idx == rank:
                patch = z[:, :, :,
                          i * p_h : (i + 1) * p_h,
                          j * p_w : (j + 1) * p_w]
                decoded = self._decode(patch)
```

```
                local_patches.append(decoded.reshape(-1))
        # 通过 all_gather 收集所有 patch 后拼接
        # ... 使用 _process_parallel_tiled_outputs 的变体 ...

    def _process_parallel_tiled_outputs(self, ...):
        """
        通用并行 tile/patch 输出处理：
        1. 每个 rank 将本地结果填充到相同大小
        2. gather 到 rank 0
        3. 按 tile/patch 索引合并并 blend
        """
        # ... 实现集中了原有的 gather/merge 逻辑
```

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

新增的自定义算子模块，封装 Fusion LayerNorm + Scale/Shift + Gate 操作，提供 CUDA Triton 和通用 fallback

```
# python/sglang/multimodal_gen/runtime/layers/fused_scale_shift_gate.py
# SPDX-License-Identifier: Apache-2.0

from typing import Optional, Tuple
import torch
import torch.nn.functional as F

from sglang.multimodal_gen.runtime.layers.custom_op import CustomOp
from sglang.multimodal_gen.runtime.platforms import current_platform

# 只在 CUDA 下导入 Triton kernel，避免在 HIP/CPU 上编译错误
_is_cuda = current_platform.is_cuda()
if _is_cuda:
    from sglang.jit_kernel.diffusion.triton.scale_shift import (
        fuse_layernorm_scale_shift_gate_select01_kernel,
        fuse_residual_layernorm_scale_shift_gate_select01_kernel,
    )


@CustomOp.register("fuse_layernorm_scale_shift_gate_select01")
class FusedLayerNormScaleShiftGateSelect01(CustomOp):
    """Fused layernorm + scale/shift + gate with binary index selection.
    CUDA path uses a Triton kernel; other platforms fall back to PyTorch ops.
    """

    def forward_cuda(self, x, weight, bias, scale0, shift0, gate0,
                     scale1, shift1, gate1, index, eps):
        # 保证输入连续
        x = x.contiguous()
        index = index.contiguous()
        return fuse_layernorm_scale_shift_gate_select01_kernel(
            x, weight=weight, bias=bias,
```

```
            scale0=scale0.contiguous(), shift0=shift0.contiguous(),
            gate0=gate0.contiguous(),
            scale1=scale1.contiguous(), shift1=shift1.contiguous(),
            gate1=gate1.contiguous(),
            index=index, eps=eps,
        )

    def forward_hip(self, *args, **kwargs):
        # HIP 平台暂时使用 native fallback
        return self.forward_native(*args, **kwargs)

    def forward_native(self, x, weight, bias, scale0, shift0, gate0,
                       scale1, shift1, gate1, index, eps):
        idx = index.to(dtype=torch.bool).unsqueeze(-1)
        shift = torch.where(idx, shift1.unsqueeze(1), shift0.unsqueeze(1))
        scale = torch.where(idx, scale1.unsqueeze(1), scale0.unsqueeze(1))
        gate = torch.where(idx, gate1.unsqueeze(1), gate0.unsqueeze(1))
        x = F.layer_norm(x, (x.shape[-1],), weight=weight, bias=bias, eps=eps)
        x = x * (1 + scale) + shift
        return x, gate


@CustomOp.register("fuse_residual_layernorm_scale_shift_gate_select01")
class FusedResidualLayerNormScaleShiftGateSelect01(CustomOp):
    # 类似，增加了 residual 和 residual_gate 参数
    # ...（代码结构相同，加 residual 分支）
```

评论区精华

Gemini Code Assist 提出了 3 条性能优化建议：

- 在 common.py 的 patch 和 tiled 解码中，使用 torch.zeros 直接分配 gather buffer，避免 zeros_like + repeat 的中间张量拷贝。
- 改用 tensor-based all_gather 替代 all_gather_object，减少 pickle 序列化开销。

这些评论未收到回复，但 PR 已合并，可能已内部处理或留作后续优化。

- 优化 gather buffer 分配方式 (performance): 未收到回复，PR 已合并，该优化未在本次 PR 中采纳。
- 类似优化提议（tiled decode 路径） (performance): 未回复，未采纳。
- 使用 tensor-based all_gather 替代 all_gather_object (performance): 未回复，评论指出当前可接受，但若成为瓶颈可后续优化。

风险与影响

- 风险：
  1. 兼容性：新增的 parallel_decode_mode 配置被默认关闭（use_parallel_decode=False），不影响已有行为，但新增的 fused 算子依赖 CustomOp 注册，如果其他模型也使用了

类似的 scale/shift 操作但未通过该算子，可能产生冲突。

2. 性能：parallel_patch_decode 和 parallel_tiled_decode 中的 all_gather 和 gather 为同步通信，在 GPU 数较多时可能成为瓶颈。
3. 精度：parallel_decode_mode="auto" 的阈值判断基于 latent 尺寸与 tile 最小尺寸比较，对于边界情况可能选择非最优路径。
4. 测试覆盖：PR 未包含自动化测试，仅提供了手工精度和速度对比，新的 fused 算子在非 CUDA 平台（NPU / AMD）上的验证依赖原生 fallback，可能缺少充分测试。 - 影响：用户：Qwen-Image 用户可通过 CLI 参数启用并行 VAE 解码并选择策略，大图使用 tiled 模式，小图使用 patch 模式以平衡性能与显存；默认关闭，无 break。系统：改进后，VAE 解码在多 GPU 环境下可更灵活地分配计算，减少单卡显存压力（如 1k 图显存从 49.75GB 降至 41.65GB），解码速度在 patch 模式下也优于 base。团队：代码结构更清晰，fused 算子统一了不同平台的实现路径，便于后续为 NPU/AMD 添加专用 kernel。

- 风险标记：核心解码路径变更，缺少自动化测试，新算子平台兼容性依赖 fallback，auto 模式边界决策风险

关联脉络

- PR #23736 [Diffusion] Refactor CFG Parallelism Framework to Support Multi-branch CFG for LTX2 Models: 同属 diffusion 模块的并行优化重构，涉及 pipeline stages 和分布式工具，与本 PR 的 VAE 并行策略形成互补，但无直接依赖。
- PR #24494 [diffusion] Precompute LTX2 guidance perturbation states: 同为 diffusion 性能优化，预计算策略。与本 PR 的 patch 并行解码可能共同用于提升推理效率。