PR #29777 完整报告

sgl-project/sglang

[diffusion] Support SP for Krea-2

合并时间：2026-07-08 10:39

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/29777

执行摘要

- 一句话：为 Krea-2 添加序列并行（Ulysses）支持
- 推荐动作：此 PR 值得精读，特别是 _forward_with_replicated_prefix 的 GQA 修复和 _shard_img_pos_for_sp 的分片逻辑，展示了如何在扩散注意力中正确实现 Ulysses 序列并行。设计决策（replicated text prefix + sharded image tokens）对类似架构有参考价值。

功能与动机

Krea-2 的单流 MMDiT 已经使用 USPAttention，但缺乏序列并行感知，导致 --ulysses-degree 产生重复 / 平铺的图像。通过添加 SP 支持，可以在多 GPU 上获得近 1.7×/2.8× 的 denoise 加速（2 GPU/4 GPU），并且 SP 输出与单 GPU 位元一致。

实现拆解

1. 修复 USPAttention replicated-prefix 的 GQA 头部切片：在 _forward_with_replicated_prefix 中，为 K/V 使用独立的 kv_h_local 分片偏移，而非复用 Q 的分片，避免 GQA 下 K/V 切片为空或不匹配。
2. 在 Krea-2 模型中传播 num_replicated_prefix：在 krea2.py 的 AttentionBlock.forward 和 Krea2DitModel._forward_impl 中添加 num_replicated_prefix 和 skip_sequence_parallel 参数，让文本前缀（txtlen）不被 all-to-all 切分。
3. 添加图像 RoPE 位置分片：在 pipeline_configs/krea2.py 的 _build_pos_and_mask 中，当 sp_world_size > 1 时，使用 _shard_img_pos_for_sp 对图像位置进行 padding 和分片，并设置 mask=None；同时限制 SP 下仅支持单 prompt（ragged 批次会引发 ValueError）。
4. 新增单元测试：test_usp_attention_replicated_prefix.py 使用 mock 模拟多 rank 环境，验证 GQA（8 query/2 KV heads）和 MHA（8 query/8 KV heads）下 replicated prefix 的 shape 正确性。
5. 更新文档：在 Krea-2.mdx 中添加多 GPU 章节，说明 TP、SP 及混合使用的方法和 trade-off，并提供命令行示例。

关键文件：

- python/sglang/multimodal_gen/test/unit/test_usp_attention_replicated_prefix.py（模块测试；类别 test；类型 test-coverage；符号 _fake_input_all_to_all, _fake_output_all_to_all, _CaptureAttn, init）：新增的单元测试，验证 USPAttention

replicated-prefix 在 GQA 和 MHA 下的 head sharding 正确性，是修复的回归保障。

- python/sglang/multimodal_gen/runtime/models/dits/krea2.py（模块 模型；类别 source；类型 core-logic；符号 AttentionBlock.forward, Krea2DitModel._forward_impl, USPAttention.forward）：Krea-2 模型核心实现，修改了 AttentionBlock.forward 和 Krea2DitModel._forward_impl 以传播 num_replicated_prefix，使文本前缀绕过 all-to-all。
- python/sglang/multimodal_gen/configs/pipeline_configs/krea2.py（模块 配置；类别 source；类型 dependency-wiring；符号 _build_pos_and_mask, _shard_img_pos_for_sp）：Krea-2 管道配置，添加了图像 RoPE 位置分片逻辑 _shard_img_pos_for_sp，并限制 SP 下只支持单 prompt。
- python/sglang/multimodal_gen/runtime/layers/attention/layer.py（模块 注意力层；类别 source；类型 core-logic；符号 _forward_with_replicated_prefix）：注意力层核心修复：使 _forward_with_replicated_prefix 在 GQA 下正确分片 K/V 前缀，使用独立的 kv_h_local 偏移。
- docs_new/cookbook/diffusion/Krea/Krea-2.mdx（模块 文档；类别 other；类型 documentation）：文档更新，添加多 GPU 章节，说明 TP、SP 及混合配置，提供命令行示例和 trade-off 分析。

关键符号：_forward_with_replicated_prefix, _shard_img_pos_for_sp, _build_pos_and_mask, AttentionBlock.forward, Krea2DitModel._forward_impl, test_gqa_slices_kv_prefix_by_kv_heads

关键源码片段

python/sglang/multimodal_gen/test/unit/test_usp_attention_replicated_prefix.py

新增的单元测试，验证 USPAttention replicated-prefix 在 GQA 和 MHA 下的 head sharding 正确性，是修复的回归保障。

```
"""Regression test for USPAttention GQA replicated-prefix head sharding.

For GQA models (kv heads < q heads), the K/V prefix must be sliced by the KV head
shard instead of the query head shard, otherwise the prefix slice is empty and cat
with the all-to-all'd suffix raises. MHA (kv heads == q heads) is unaffected.
"""

import unittest
from unittest.mock import MagicMock, patch

import torch

from sglang.multimodal_gen.runtime.layers.attention.layer import USPAttention

_LAYER = "sglang.multimodal_gen.runtime.layers.attention.layer"
_SP = 2 # Ulysses world size

def _fake_input_all_to_all(x, **_):
```

```
    # Simulate Ulysses input all-to-all: gather sequence (xSP), shard heads (/SP).
    h = x.shape[2]
    return x[:, :, : h // _SP, :].repeat_interleave(_SP, dim=1).contiguous()


def _fake_output_all_to_all(x, **_):
    # Inverse: shard sequence (/SP), gather heads (xSP).
    s = x.shape[1]
    return x[:, : s // _SP, :, :].repeat_interleave(_SP, dim=2).contiguous()


class _CaptureAttn:
    """Stand-in attention backend that records the q/k/v it receives."""

    def __init__(self):
        self.q = self.k = self.v = None

    def forward(self, q, k, v, _ctx):
        self.q, self.k, self.v = q, k, v
        return q.clone()


class TestUSPAttentionReplicatedPrefix(unittest.TestCase):
    def _run(self, q_heads, kv_heads, sp_rank, num_rep=3, suffix=4, head_dim=4):
        attn = _CaptureAttn()
        # Bypass __init__/backend setup; directly set attn_impl
        obj = USPAttention.__new__(USPAttention)
        obj.attn_impl = attn

        seq = num_rep + suffix
        q = torch.randn(1, seq, q_heads, head_dim)
        k = torch.randn(1, seq, kv_heads, head_dim)
        v = torch.randn(1, seq, kv_heads, head_dim)

        sp_group = MagicMock()
        sp_group.ulysses_group = None

        def fake_all_gather(out_list, tensor, **_):
            for t in out_list:
                t.copy_(tensor)

        with (
            patch(f"{_LAYER}.get_ulysses_parallel_world_size", return_value=_SP),
            patch(f"{_LAYER}.get_sp_parallel_rank", return_value=sp_rank),
            patch(f"{_LAYER}._usp_input_all_to_all", side_effect=_fake_input_all_to_all),
            patch(f"{_LAYER}._usp_output_all_to_all", side_effect=_fake_output_all_to_all),
            patch(f"{_LAYER}.get_sp_group", return_value=sp_group),
            patch("torch.distributed.all_gather", side_effect=fake_all_gather),
        ):
```

```
            out = USPAttention._forward_with_replicated_prefix(obj, q, k, v, None, num_rep)
        return attn, out, q.shape

    def test_gqa_slices_kv_prefix_by_kv_heads(self):
        # GQA: 8 query heads, 2 kv heads. Old code sliced by query heads,
        # causing empty K/V prefix. New code uses kv_h_local.
        for sp_rank in range(_SP):
            with self.subTest(sp_rank=sp_rank):
                attn, out, q_shape = self._run(q_heads=8, kv_heads=2, sp_rank=sp_rank)
                # q keeps q_heads/SP, k/v keep kv_heads/SP -> GQA grouping preserved
                self.assertEqual(attn.q.shape[2], 8 // _SP)
                self.assertEqual(attn.k.shape[2], 2 // _SP)
                self.assertEqual(attn.v.shape[2], 2 // _SP)
                # prefix + all-to-all'd suffix line up on the sequence axis
                self.assertEqual(attn.k.shape[1], attn.q.shape[1])
                # output is restored to the input layout
                self.assertEqual(tuple(out.shape), tuple(q_shape))
```

python/sglang/multimodal_gen/configs/pipeline_configs/krea2.py

Krea-2 管道配置，添加了图像 RoPE 位置分片逻辑 _shard_img_pos_for_sp，并限制 SP 下只支持单 prompt。

```
# Excerpt from Krea2PipelineConfig._build_pos_and_mask and _shard_img_pos_for_sp


def _build_pos_and_mask(self, context, text_mask, batch, device):
    b, txt_len = context.shape[0], context.shape[1]
    patch = self.dit_config.arch_config.patch
    vsf = self.get_vae_scale_factor()
    h_tok = int(batch.height) // vsf // patch
    w_tok = int(batch.width) // vsf // patch

    img_ids = torch.zeros(h_tok, w_tok, 3, device=device)
    img_ids[..., 1] = torch.arange(h_tok, device=device)[:, None]
    img_ids[..., 2] = torch.arange(w_tok, device=device)[None, :]
    img_pos = img_ids.reshape(h_tok * w_tok, 3).unsqueeze(0).expand(b, -1, -1)
    txt_pos = torch.zeros(b, txt_len, 3, device=device)

    sp_world_size = get_sp_world_size()
    if sp_world_size > 1:
        # Under SP: shard image positions to match latent sharding, text stays replicated.
        # Ragged/padded batches are not supported.
        if text_mask is not None and not bool(text_mask.all()):
            raise ValueError(
                "Krea-2 sequence parallelism does not support ragged/padded "
                "multi-prompt batches; use a single prompt or --tp-size."
            )
        img_pos = self._shard_img_pos_for_sp(img_pos, sp_world_size)
        return {"pos": torch.cat([txt_pos, img_pos], dim=1), "mask": None}
```

```
        # Non-SP path: build full mask as before.
        pos = torch.cat([txt_pos, img_pos], dim=1)
        img_mask = torch.ones(b, h_tok * w_tok, dtype=torch.bool, device=device)
        if text_mask is None:
            txt_mask = torch.ones(b, txt_len, dtype=torch.bool, device=device)
        else:
            txt_mask = text_mask.to(device=device).bool()
        mask = torch.cat([txt_mask, img_mask], dim=1)
        return {"pos": pos, "mask": mask}


    @staticmethod
    def _shard_img_pos_for_sp(img_pos, sp_world_size):
        # Pad the image-position sequence to a multiple of sp_world_size,
        # then take this rank's contiguous slice.
        s = img_pos.shape[1]
        if s % sp_world_size != 0:
            pad = img_pos[:, -1:].repeat(1, sp_world_size - (s % sp_world_size), 1)
            img_pos = torch.cat([img_pos, pad], dim=1)
        local = img_pos.shape[1] // sp_world_size
        rank = get_sp_parallel_rank()
        return img_pos[:, rank * local : (rank + 1) * local]
```

评论区精华

没有公开的 review 评论。两位 reviewer（mickqian, zijiexia）均批准，无争议。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. GQA 修复影响：修复可能导致其他依赖 _forward_with_replicated_prefix 的模型（如未来扩散模型）的行为变化，但 MHA 模型不受影响（kv_h_local == h_local）。
  2. SP 路径限制：SP 当前不支持 ragged/padded 多 prompt 批次，用户需确保单 prompt 或退回到 TP。
  3. 性能风险：当 sp_world_size 为 1 时，新分支不会触发，性能无影响。
  4. 测试覆盖：新增单元测试覆盖了核心 slicing 逻辑的 shape 正确性，但未覆盖实际分布式场景端到端。 - 影响：用户现在可以使用 --ulysses-degree N 参数对 Krea-2 进行多 GPU 推理，获得接近线性的加速（2 GPU 1.71× denoise 加速），同时保持 bitwise 精确输出。需要至少 2 个 GPU。TP 和 SP 可组合，提供灵活配置。SP 路径目前限制为单 prompt。 - 风险标记：GQA 修复潜在回归，SP 路径单 prompt 限制，测试未覆盖端到端分布式

关联脉络

- PR #29742 [diffusion] Fix Z-Image accuracy: 同属 diffusion 模块，且修改了 attention/layer.py，与本 PR 的注意力层修复可能有重叠。
- PR #30150 [diffusion][cache-dit] add dual-transformer Cache-DiT adapter specs: 同样修改 diffusion 管道配置和注意力层，展示了 diffusion 子系统的演化脉络。