PR #33923 完整报告

sgl-project/sglang

[Diffusion] Route zimage and hunyuanvideo attention through USPAttention

合并时间：2026-08-07 12:36

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33923

执行摘要

- 一句话：zimage 与 hunyuanvideo 注意力迁移至 USPAttention，suffix 位级稳定
- 推荐动作：建议精读。这是 diffusion 分布式注意力收敛的关键一步：layer.py 的 _forward_with_replicated_suffix 顺序保持实现是“SP 下复制 token 处理”的范本；“fail loudly 而不是静默错误”的守卫设计值得借鉴；parity 测试的契约划分（suffix bitwise、prefix ≤1e-2）是数值稳定性测试的好例子。后续可关注 ring 并行下复制路径的支持。

功能与动机

PR body 明确目标是迁移剩余三个 stacked-QKV UlyssesAttention 调用点（zimage 的 replicated-suffix 分支、hunyuanvideo 的 double-stream 与 single-stream 块）到 USPAttention，让 UlyssesAttention 不再有模型调用点，从而解锁这些路径的 ring 支持。另一个动机是数值稳定性：旧 suffix 实现把复制 token 旋转到序列前部以复用 prefix 路径，数学上正确但重排了每个 query 的 K/V 扫描顺序，少步数（turbo）模型会把 bf16 重排放大成可见漂移（Z-Image-Turbo 9 步 MAE 7.28、30 步 MAE 6.32 不收敛），而顺序保持实现为 0（bitwise）。

实现拆解

1. 补强 USPAttention（layer.py）：forward 新增 seq_lens 参数，并把 UlyssesAttention 的 stacked varlen 实现逐行移植为独立分支（拼接 QKV 后经 _usp_input_all_to_all_varlen 交换、preprocess_qkv/postprocess_output 包装），保证 hunyuanvideo sharded 文本分支语义不变；varlen 分支显式断言不接受 mask 与复制 token。_forward_with_replicated_suffix 从“旋转到前部复用 prefix 路径”改为“顺序保持”实现：分片段走 Ulysses all-to-all，复制段按 rank 切 head 后拼在序列尾部，输出端分片段走 all-to-all、复制段走显式 all_gather，并新增 ring 并行不支持的保护。masked 路径入口新增 NotImplementedError 守卫，拒绝 replicated 参数。
2. zimage 迁移（zimage.py）：删除 ulysses_attn 属性与 num_replicated_suffix 特判分支（约 47 行），注意力一律走 self.attn（USPAttention），replicated suffix 由新的顺序保持路径处理。
3. hunyuanvideo 迁移（hunyuanvideo.py）：MMDoubleStreamBlock 与 MMSingleStreamBlock 的 UlyssesAttention 换成 USPAttention；sharded 文本分支改用 seq_lens varlen 入口，非 sharded 分支把文本拼到尾部并用 num_replicated_suffix，返回值从 tuple 收敛为单 tensor，统一在调用侧 split 回 img 与 txt。

4. 测试配套：新增 test_usp_replicated_parity_2_gpu.py（2 GPU 双进程 parity：suffix 逐位相等、prefix 漂移 ≤1e-2），在 gpu_cases.py 的 2-gpu standalone 列表注册并给出 180 秒耗时估算；test_usp_attention_replicated_prefix.py 补充 masked 守卫单测。

关键文件：

- python/sglang/multimodal_gen/runtime/layers/attention/layer.py（模块 注意力层；类别 source；类型 core-logic；符号 USPAttention.forward, USPAttention._forward_with_replicated_suffix）：USPAttention 核心实现：新增 seq_lens varlen 入口、重写 replicated suffix 顺序保持路径、masked 分支守卫，是整个迁移的地基。
- python/sglang/multimodal_gen/runtime/models/dits/zimage.py（模块 扩散模型；类别 source；类型 data-contract；符号 ZImageAttention.forward, ZImageAttention.init）：移除最后一个 stacked-QKV UlyssesAttention 分支与 ulysses_attn 属性，replicated suffix 统一走 USPAttention，Z-Image 多 GPU 数值位级稳定。
- python/sglang/multimodal_gen/runtime/models/dits/hunyuanvideo.py（模块 扩散模型；类别 source；类型 data-contract；符号 MMDoubleStreamBlock.forward, MMSingleStreamBlock.forward）：double-stream 与 single-stream 两个块迁移到 USPAttention；sharded 文本用 varlen，非 sharded 文本用 num_replicated_suffix，行为与 legacy 一致（md5 相同）。
- python/sglang/multimodal_gen/test/single_test_file/test_usp_replicated_parity_2_gpu.py（模块 双卡测试；类别 test；类型 test-coverage；符号 _worker, Sdpa, TestUSPReplicatedParity, test_replicated_parity_two_ranks）：新增 2 GPU parity 契约测试：suffix 位级相等、prefix 漂移 ≤1e-2，防止顺序保持实现回归。
- python/sglang/multimodal_gen/test/unit/test_usp_attention_replicated_prefix.py（模块 单元测试；类别 test；类型 test-coverage；符号 TestUSPAttentionMaskedReplicatedGuard, test_masked_path_rejects_replicated_tokens）：新增 masked 路径拒绝复制 token 的守卫单测，覆盖新增 NotImplementedError 行为。
- python/sglang/multimodal_gen/test/server/gpu_cases.py（模块 测试注册；类别 test；类型 test-coverage）：把新 parity 测试注册到 2-gpu standalone 用例并提供 CI 耗时估算。

关键符号：USPAttention.forward, USPAttention._forward_with_replicated_suffix, ZImageAttention.forward, MMDoubleStreamBlock.forward, MMSingleStreamBlock.forward, TestUSPReplicatedParity.test_replicated_parity_two_ranks, TestUSPAttentionMaskedReplicatedGuard.test_masked_path_rejects_replicated_tokens

关键源码片段

python/sglang/multimodal_gen/runtime/layers/attention/layer.py

USPAttention 核心实现：新增 seq_lens varlen 入口、重写 replicated suffix 顺序保持路径、masked 分支守卫，是整个迁移的地基。

USPAttention._forward_with_replicated_suffix 是本次迁移的核心：把复制后缀保留在序列尾部，绕过 all-to-all，保证与单 rank 完全相同的 K/V 扫描顺序。

```
  def _forward_with_replicated_suffix(
      self,
      q: torch.Tensor,
      k: torch.Tensor,
      v: torch.Tensor,
      ctx_attn_metadata,
      num_rep: int,
  ) -> torch.Tensor:
      """Ulysses 注意力下，序列尾部 num_rep 个 token 在每个 SP rank 上完全复制。

      这些复制 token 不应再经过 all-to-all 被重复，而是保留在序列尾部。
      这样每个 query 以与单 rank 完全相同的顺序扫描 K/V，跨 SP 度保持
      bitwise 稳定；若旋转到序列头部，归约顺序被重排，少步数（turbo）
      模型会把 bf16 重排放大为可见漂移。
      """
      if num_rep <= 0:
          raise ValueError("num_rep must be positive for replicated suffix.")
      if get_ring_parallel_world_size() > 1:
          # 复制前缀 / 后缀路径尚未支持 ring 并行，明确拒绝比静默出错更好。
          raise NotImplementedError(
              "USPAttention replicated-prefix/suffix path does not support "
              "ring parallelism yet."
          )
      sp_rank = get_sp_parallel_rank()

      # 将复制段（尾部）与分片段切开，分片段走 Ulysses all-to-all。
      q_shard, q_rep = q[:, :-num_rep], q[:, -num_rep:]
      k_shard, k_rep = k[:, :-num_rep], k[:, -num_rep:]
      v_shard, v_rep = v[:, :-num_rep], v[:, -num_rep:]

      q_shard = _usp_input_all_to_all(q_shard, head_dim=2)
      k_shard = _usp_input_all_to_all(k_shard, head_dim=2)
      v_shard = _usp_input_all_to_all(v_shard, head_dim=2)

      h_local = q_shard.shape[2] # 本 rank 的 Q head 数
      kv_h_local = k_shard.shape[2] # 本 rank 的 KV head 数
      h_start = sp_rank * h_local
      kv_h_start = sp_rank * kv_h_local
      # 复制段也按 rank 切分 head，保证拼接后每个 head 恰好出现一次。
      q_rep = q_rep[:, :, h_start : h_start + h_local, :].contiguous()
      k_rep = k_rep[:, :, kv_h_start : kv_h_start + kv_h_local, :].contiguous()
      v_rep = v_rep[:, :, kv_h_start : kv_h_start + kv_h_local, :].contiguous()

      # 顺序保持：分片在前、复制段在后，与单 rank 序列顺序一致。
      q = torch.cat([q_shard, q_rep], dim=1)
      k = torch.cat([k_shard, k_rep], dim=1)
```

```
v = torch.cat([v_shard, v_rep], dim=1)

out = self.attn_impl.forward(q, k, v, ctx_attn_metadata)

out_shard = out[:, :-num_rep]
out_rep = out[:, -num_rep:]

out_shard = _usp_output_all_to_all(out_shard, head_dim=2)

# 复制段输出在 head 维上 all-gather，恢复完整 head 数。
sp_size = get_ulysses_parallel_world_size()
gathered = [torch.empty_like(out_rep) for _ in range(sp_size)]
torch.distributed.all_gather(
    gathered,
    out_rep.contiguous(),
    group=get_sp_group().ulysses_group,
)
out_rep = torch.cat(gathered, dim=2)

return torch.cat([out_shard, out_rep], dim=1)
```

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

double-stream 与 single-stream 两个块迁移到 USPAttention；sharded 文本用 varlen，非 sharded 文本用 num_replicated_suffix，行为与 legacy 一致（md5 相同）。

以 double-stream 块为例，非 sharded 文本分支用 num_replicated_suffix 表示文本 token 作为复制后缀。

```
# 文本未分片时，文本 token 位于序列尾部且各 rank 完全一致：
# 用 num_replicated_suffix 让 USPAttention 跳过文本段的 all-to-all，
# 并保持与单 rank 完全相同的 K/V 扫描顺序（bitwise 稳定）。
if txt_is_sharded:
    attn = self.attn(
        torch.cat((img_q, txt_q), dim=1),
        torch.cat((img_k, txt_k), dim=1),
        torch.cat((img_v, txt_v), dim=1),
        seq_lens=seq_lens, # varlen 语义：保留 UlyssesAttention 行为
    )
else:
    attn = self.attn(
        torch.cat((img_q, txt_q), dim=1),
        torch.cat((img_k, txt_k), dim=1),
        torch.cat((img_v, txt_v), dim=1),
        num_replicated_suffix=text_seq_len,
    )
img_attn, txt_attn = attn.split([image_seq_len, text_seq_len], dim=1)
```

python/sglang/multimodal_gen/test/single_test_file/test_usp_replicated_parity_2_gpu.py

新增 2 GPU parity 契约测试：suffix 位级相等、prefix 漂移 ≤1e-2，防止顺序保持实现回归。

parity 测试核心校验：suffix 要求 bitwise 相等，prefix 允许有界漂移。

```
# suffix：复制段在尾部，输出必须与单 rank 参考逐位一致。
out = attn.forward(
    torch.cat([qf[:, sl], qf[:, S:]], dim=1),
    torch.cat([kf[:, sl], kf[:, S:]], dim=1),
    torch.cat([vf[:, sl], vf[:, S:]], dim=1),
    num_replicated_suffix=REP,
)
exp = torch.cat([ref[:, sl], ref[:, S:]], dim=1)
if not torch.equal(out, exp):
    d = (out.float() - exp.float()).abs()
    failures.append(f"suffix not bitwise: mae={d.mean():.3e} max={d.max():.3e}")

# prefix：允许有界漂移（≤1e-2），因为前缀路径会改变归约顺序。
out_p = attn.forward(
    torch.cat([qf[:, S:], qf[:, sl]], dim=1),
    torch.cat([kf[:, S:], kf[:, sl]], dim=1),
    torch.cat([vf[:, S:], vf[:, sl]], dim=1),
    num_replicated_prefix=REP,
)
exp_p = torch.cat([ref[:, S:], ref[:, sl]], dim=1)
dp = (out_p.float() - exp_p.float()).abs()
if dp.max().item() > 1e-2:
    failures.append(f"prefix drift: mae={dp.mean():.3e} max={dp.max():.3e}")
```

评论区精华

该 PR 没有实质 review 评论（仓库仅有一条 /tag-and-rerun-ci 的 CI 重跑指令），核心技术讨论体现在 PR body 与提交说明中：

- rotate-to-front 与 order-preserving 的取舍：PR body 给出实测数据，rotate-to-front 在 Z-Image-Turbo 上 9 步 MAE 7.28、30 步 6.32（不收敛），order-preserving 为 0（bitwise）。单层 fp32 MAE 仅 2e-8，说明函数在隔离状态下数值正确，但 bf16 重排的微小差异被 turbo 类模型放大，因此选择顺序保持并固化为契约。
- masked 路径与复制 token 的组合：commit 4 说明 masked 分支把每一行都送进 all-to-all，从未消费 num_replicated_prefix/suffix/kv_prefix，replicated 段会在各 rank 被重复并静默损坏输出，因此选择 fail loudly 抛 NotImplementedError。
  - 复制后缀旋转到前部 vs 顺序保持 (design): 采用顺序保持实现，后缀留在序列尾部；新增 parity 测试把 suffix bitwise、prefix 漂移 ≤1e-2 固化为契约。
  - masked 路径与复制 token 的组合 (correctness): 在 masked 分支入口增加 NotImplementedError 守卫，并补单测。

风险与影响

- 风险：

1. 核心注意力路径重写：layer.py 的 _forward_with_replicated_suffix 是完全重写的实现，涉及 all-to-all、显式 all_gather 与 head 维 slicing；2 GPU parity 测试只覆盖纯 SDPA 后端，未覆盖 FA/SageAttention 与真实模型端到端（md5 一致性为手工验证），存在回归窗口。
2. masked 路径行为变更：此前 replicated + mask 会静默产生错误输出，现在直接抛 NotImplementedError；若下游存在依赖旧行为（错误但未暴露）的调用会被中断，这是有意为之但仍是行为变更。
3. varlen 入口限制：seq_lens 分支断言不接受 mask 或任何 replicated 参数，未来若需要组合使用会受限制。
4. ring 并行受限：replicated 路径在 get_ring_parallel_world_size() > 1 时抛异常，zimage/hunyuanvideo 在 ring > 1 时无法使用 num_replicated_suffix；当前这两个模型 ring 默认 1，风险可控。
5. 性能影响：复制段输出走显式 all_gather，且顺序保持可能削弱 kernel 局部性；复制 token 量小（文本 / 条件 token），影响可忽略。

- 影响：
  1. 用户 / 模型侧：diffusion 多 GPU（SP/ulysses）推理下，Z-Image 与 HunyuanVideo 的输出与单 GPU 位级一致，turbo 少步数模型不再出现像素漂移，多卡结果可靠性显著提升。
  2. 系统 / 架构侧：diffusion 注意力后端收敛到 USPAttention 单一入口，UlyssesAttention 仅保留 UlyssesAttention_VSA 基类与 IPC 测试载体；后续 ring support 只需扩展 USPAttention。
  3. 团队 /CI 侧：新增 2 GPU parity 测试（约 180 秒）进入 CI，将“复制段位级一致”固化为回归门槛；masked 路径由静默错误改为显式异常，可能暴露此前被掩盖的调用问题。 - 风险标记：核心注意力路径重写，masked 路径行为变更，ring 并行下复制路径受限，缺少真实模型端到端 CI 断言

关联脉络

- PR #32667 [Diffusion] Add K/V-gather sequence parallel attention: 同一 USPAttention / SP 注意力演进线，均以 layer.py 为核心；本次迁移为 ring 支持铺路。
- PR #33823 [diffusion] FLUX.2 bit-exact residual-gate fast path (H200 klein-4B 50-step denoise -1.2%): 同属 bitwise 数值稳定主题，体现 diffusion 模块对位级一致性的重视。
- PR #33848 [diffusion] resolve IPC A2A peers from process groups: 同属 diffusion 分布式通信路径修复，共同收敛 diffusion 多卡基础设施。