PR #29147 完整报告

sgl-project/sglang

[diffusion][perf]: Shard Qwen Text Embed in SP

合并时间：2026-06-26 21:44

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/29147

执行摘要

- 一句话：对 Qwen Image 文本嵌入进行序列并行分片，降低显存和 GEMM 计算，提速约 3%
- 推荐动作：值得合并。关键设计决策是将分片分为均匀和非均匀两条路径，避免强制填充带来的额外计算。建议在后续 PR 中考虑添加 .contiguous() 以增强鲁棒性，并补充单元测试覆盖边界条件。

功能与动机

当前 Qwen Image 未对文本嵌入进行序列并行分片，而是在所有 GPU 上复制，虽然节省了通信但增加了 GEMM 计算量。借鉴 FLUX 的类似优化 PR #27066，本 PR 对文本嵌入进行分片，以降低 GEMM 开销。

实现拆解

1. 新增 _shard_text_for_sp 函数：当文本序列长度可被 SP world size 整除时，使用 torch.chunk 将文本嵌入和 RoPE cache 沿序列维度均匀分配给各 rank。
2. 新增 _pad_shard_text_for_sp_varlen 函数：处理不可整除的情况，先右填充文本嵌入和 RoPE cache 至可被整除，再分片，同时生成 attn_mask 和 attn_mask_meta 供 USPAttention 的 varlen kernel 忽略填充 token。
3. 修改模型前向逻辑：在 QwenImageDitBlock 中根据文本长度是否整除选择调用上述函数，并设置 num_replicated_prefix=0 启用 fully sequence-parallel joint attention。
4. 更新导入：添加 get_ring_parallel_world_size 和 get_sp_parallel_rank 依赖。

关键文件：

- python/sglang/multimodal_gen/runtime/models/dits/qwen_image.py（模块 扩散模型；类别 source；类型 core-logic；符号 _shard_text_for_sp, _pad_shard_text_for_sp_varlen）：核心实现文件，新增两个分片函数并修改前向逻辑进行分片调用。

关键符号：_shard_text_for_sp, _pad_shard_text_for_sp_varlen

关键源码片段

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

核心实现文件，新增两个分片函数并修改前向逻辑进行分片调用。

```
def _shard_text_for_sp(
    encoder_hidden_states: torch.Tensor,
    freqs_cis: Optional[Tuple[torch.Tensor, torch.Tensor]],
) -> Tuple[torch.Tensor, Optional[Tuple[torch.Tensor, torch.Tensor]]]:
    # 如果 SP size 为 1 则不进行任何分片
    sp_size = get_sp_world_size()
    if sp_size == 1:
        return encoder_hidden_states, freqs_cis

    # 将文本嵌入沿序列维度（dim=1）均匀切分并取当前 rank 的分片
    sp_rank = get_sp_parallel_rank()
    encoder_hidden_states = torch.chunk(encoder_hidden_states, sp_size, dim=1)[sp_rank]

    if freqs_cis is not None:
        img_cache, txt_cache = freqs_cis
        # 同样对文本 RoPE cache 进行分片（沿 dim=0）
        txt_cache = torch.chunk(txt_cache, sp_size, dim=0)[sp_rank]
        freqs_cis = (img_cache, txt_cache)

    return encoder_hidden_states, freqs_cis
```

varlen 路径函数 _pad_shard_text_for_sp_varlen 类似，但先右填充再分片，并返回 attn_mask 和 attn_mask_meta。

评论区精华

Gemini Code Assist 机器人建议在 torch.chunk 后添加 .contiguous() 以避免自定义 CUDA 内核因非连续张量出错或性能下降。作者回复测试未发现错误，未采纳建议。

- 分片后张量连续性 (performance): 作者回复测试未出现错误，未添加 .contiguous()。

风险与影响

- 风险：数值精度与视觉输出经测试无差异，但 varlen 路径引入的填充和掩码逻辑存在边界条件风险；解码阶段延迟增加约 20%（因掩码计算），但整体获益；分片后 encoder_hidden_states 未调用 .contiguous()，可能在特定底层内核中引发隐藏错误。
- 影响：仅影响 Qwen Image 扩散模型的多 GPU 推理（ulysses degree > 1）。短提示词 E2E 延迟降低 2.6%，长提示词降低 3.4%。单 GPU 无影响。团队后续可类似优化其他复制文本嵌入的扩散模型。
- 风险标记：分片边界条件风险，varlen 路径未测试覆盖，缺少 .contiguous() 可能导致隐藏 bug

关联脉络

- 暂无明显关联 PR