

PR #36735 完整报告

sgl-project/sglang

[multimodal_gen] feat: support key masks on USPAttention's replicated-prefix path

合并时间: 2026-09-03 20:13

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/36735>

执行摘要

- 一句话: 修复 BCG + SP 下 Qwen-Image 请求因掩码拒绝而失败的问题。
- 推荐动作: 该 PR 值得精读, 特别是对于关注序列并行、BCG 和扩散模型推理的工程师。
 1. 关键设计: `_forward_with_replicated_prefix` 中对掩码进行收集以匹配序列收集的逻辑, 是处理 SP 下非均质数据 (部分复制、部分分片) 的典型模式, 具有参考价值。
 2. 测试方法: `TestUSPAttentionMaskedReplicatedPrefix` 通过精确模拟集合通信 (`fake_input_a2a`, `fake_mask_gather`) 来验证分布式逻辑的正确性, 这种测试模式值得借鉴。
 3. 上下文: 建议结合近期与 `USPAttention`、`DynamicChunkSizer`、注意力后端相关的 PR (如 #37674, #37675, #37329) 一同阅读, 以理解 SP 和 BCG 在代码库中的整体演进方向。

功能与动机

解决 breakable CUDA graphs (BCG) 在序列并行 (SP) 下对 Qwen-Image 模型完全不可用的问题。动机源自 PR body 中的直接描述: “On current main, breakable CUDA graphs combined with sequence parallelism fails every Qwen-Image request outright... The existing BCG e2e test runs `--num-gpus 1`, so CI never sees the combination.” 核心是 BCG 提示词分桶会合成一个 eager 路径不携带的 prompt mask, 而该 mask 会引导 DiT 进入其 replicated-text 分支, 导致 USPAttention 在捕获和回退阶段均拒绝执行。该 PR 通过 masked 路径服务 replicated-prefix 案例来解决此问题。

实现拆解

1. 控制流放宽与路由: 在 `USPAttention.forward` 中, 将原本对任何包含 replicated 段的 masked 路径的拒绝, 收窄为仅拒绝不合法的组合。新增 `masked_prefix_supported` 条件检查, 仅当存在 `replicated_prefix`、无 `replicated suffix/kv-prefix`、输入为 2D mask、非 `qkv_pre_all_to_all` 且非 ring 并行时, 才允许将调用路由至扩展后的 `_forward_with_replicated_prefix` 方法。
2. `_forward_with_replicated_prefix` 方法扩展: 该方法新增 `attn_mask`、`drop_masked_query_rows` 和 `joint_mask_memo_host` 参数。核心逻辑是: 在对 suffix 部分进行 all-to-all 之前, 通过 `sequence_model_parallel_all_gather` 收集 mask 的 suffix 列, 以匹配序列收集步骤。然后, 将原始 mask 的 prefix 列与收集后的 suffix 列拼接成完整的 `joint_mask`, 并在 rank-local 的注意力计算 (通过 `_replicated_kv_attention`) 中使用此联合掩码。

3. 记忆化优化：为 joint_mask 和其 varlen 元数据引入基于 BCG 重放令牌 (get_current_replay_token()) 的记忆化缓存。缓存键包含 num_rep、sp_size、mask shape 和 replay token，确保在 BCG 的不同注意力断点重放时能命中正确的缓存。缓存实例挂载在传入的 joint_mask_memo_host 对象上。
4. 新增 `_masked_local_attention` 方法：实现 rank-local 的注意力计算，镜像单 rank masked 路径。支持两种模式：当 attn_mask_meta 存在时，使用 varlen FA（通过 `_prepare_sdpa_mask` 准备的元数据）并丢弃被掩码的 query 行；否则使用带有加性掩码的 SDPA。
5. 函数提升与重用：将原先嵌套在 forward 方法内的 `_prepare_sdpa_mask` 辅助函数提升至模块作用域，使其可被新的 `_masked_local_attention` 方法重用，无行为变更。
6. 测试更新与修复：新增 `TestUSPAttentionMaskedReplicatedPrefix` 测试类，验证新路径的输出与纯全序列 masked SDPA 参考实现一致 (test_matches_full_sequence_masked_sdpa)，并断言 ring 并行和 3D mask 等非法组合仍被拒绝。同时修复了现有 `TestUSPAttentionReplicatedPrefix` 测试中因符号重命名和未补丁的 ring world size 而在主分支上失败的 mock。

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 `_prepare_sdpa_mask`, `_masked_local_attention`, `_forward_with_replicated_prefix`, `_replicated_kv_attention`）：核心逻辑变更文件，实现了对 USPAttention masked 路径的支持，包含控制流修改、新方法和缓存机制。
- 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 `TestUSPAttentionMaskedReplicatedPrefix`, `_run_rank0`, `test_matches_full_sequence_masked_sdpa`, `test_masked_prefix_still_rejects_ring_and_3d_masks`）：核心测试文件，验证了新增 masked replicated-prefix 路径的正确性，并修复了现有测试的 mock 问题。

关键符号： `_forward_with_replicated_prefix`, `_masked_local_attention`, `_prepare_sdpa_mask`, `_replicated_kv_attention`, `build_varlen_mask_meta`

关键源码片段

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

核心逻辑变更文件，实现了对 USPAttention masked 路径的支持，包含控制流修改、新方法和缓存机制。

关键源码片段：Replicated-prefix 路由与掩码收集

```
# 在 USPAttention.forward 中，新的路由条件允许合法的 masked replicated-prefix 调用
if (
    num_replicated_prefix
    or num_replicated_suffix
    or num_replicated_kv_prefix
) and not effective_skip_sp and get_sequence_parallel_world_size() > 1:
    # 判断是否为支持的 masked replicated-prefix 场景
    masked_prefix_supported = (
```

```

num_replicated_prefix > 0
and not num_replicated_suffix # 仅支持前缀复制
and not num_replicated_kv_prefix
and attn_mask is not None # 必须有掩码
and attn_mask.dim() == 2 # 支持 [B, S] 形状的 2D 掩码
and not qkv_pre_all_to_all # Ulysses-only
and get_ring_parallel_world_size() == 1 # 不支持 Ring 并行
)
if not masked_prefix_supported:
    # 保持原有的拒绝语义, 防止错误输出
    raise NotImplementedError("USPAttention's masked path does not support replicated ...")
# 将调用路由至扩展后的 replicated-prefix 处理方法
return self._forward_with_replicated_prefix(
    q, k, v, ctx_attn_metadata, num_replicated_prefix,
    attn_mask=attn_mask,
    drop_masked_query_rows=attn_mask_meta is not None,
    joint_mask_memo_host=joint_mask_memo_host,
)

```

关键源码片段: `_forward_with_replicated_prefix` 中的掩码收集与记忆化

```

def _forward_with_replicated_prefix(
    self, q, k, v, ctx_attn_metadata, num_rep: int,
    attn_mask: torch.Tensor | None = None, # 新增参数: 本地布局的 [B, num_rep + S_local] 掩码
    drop_masked_query_rows: bool = False, # 新增参数: 是否在 varlen FA 中丢弃被掩码的行
    joint_mask_memo_host=None, # 新增参数: 记忆化缓存挂载的对象
) -> torch.Tensor:
    # ... ( 分割 q/k/v 为 replicate 和 shard 部分 )
    # 对 shard 部分进行 all-to-all
    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)

    joint_mask = joint_mask_meta = None
    if attn_mask is not None:
        # 构建基于 BCG 重放令牌的缓存键, 确保在不同注意力断点重放时缓存正确
        cache_key = (num_rep, sp_size, tuple(attn_mask.shape), get_current_replay_token())
        host = joint_mask_memo_host if joint_mask_memo_host is not None else attn_mask
        cached = getattr(host, "_sglang_osp_joint_mask_cache", None)
        if cached is not None and cached[0] == cache_key:
            _, joint_mask, joint_mask_meta = cached # 命中缓存
        else:
            # 收集掩码的 suffix 列, 以匹配 suffix 序列的 all-to-all 收集
            mask_suffix = sequence_model_parallel_all_gather(
                attn_mask[:, num_rep:].contiguous(), dim=1
            )
            # 拼接 prefix 列 (在每个 rank 上相同) 和收集后的 suffix 列, 得到全序列联合掩码
            joint_mask = torch.cat([attn_mask[:, :num_rep], mask_suffix], dim=1)
            if drop_masked_query_rows:
                joint_mask_meta = build_varlen_mask_meta(joint_mask) # 为 varlen FA 准备元数据

```

```
# 将结果缓存到 host 对象上
host._sglang_usp_joint_mask_cache = (cache_key, joint_mask, joint_mask_meta)
# ... ( 后续使用 joint_mask 和 joint_mask_meta 进行注意力计算 )
```

评论区精华

PR 的 review 评论为空，讨论主要集中在 PR body 的详细说明中。核心决策点已通过代码实现：

1. 正确性确认：PR body 提供了详细的准确性测试结果（2x H200, Qwen-Image, sp=2+BCG），显示修复后的输出与 eager SP 运行结果的余弦相似度为 0.9996，且与未记忆化的 masked 路径位相等，证明了数学等价性。
 2. 性能权衡：速度测试表明，masked replicated-prefix 路径比 eager 的 sharded-text 路径稍重（需要全序列 rank-local 注意力和掩码收集），但 BCG 重放仍带来了约 7.3% 的端到端加速（3.59s -> 3.33s）。作者指出，后续可能的优化方向是将桶化文本路由通过 sharded-text 布局。
 3. 设计决策：选择支持 replicated-prefix + 2D mask 的合法组合，而非禁用 SP 下的分桶，保留了完整的分桶能力。记忆化设计（基于 replay token 的键）确保了在 BCG 下不同重放缓冲区中的掩码差异得到正确处理。
- 功能可行性与数学等价性验证 (correctness): 功能正确性已通过实验验证。
 - 性能权衡：复制路径 vs 分片路径 (performance): 当前实现是正确性优先的选择，性能已知但可接受，并为未来优化指明了方向。

风险与影响

- 风险：
 1. 核心注意力逻辑变更：_forward_with_replicated_prefix 和 _replicated_kv_attention 方法的逻辑被显著扩展。虽然测试覆盖了核心场景，但变更引入了新的条件分支和记忆化逻辑，在与复杂并行模式（如 GQA、Ring Attention）交互时可能存在未预见的边界情况。
 2. 记忆化缓存一致性：_sglang_usp_joint_mask_cache 挂载在动态的 joint_mask_memo_host 对象上。如果 BCG 重放令牌或 mask 形状的管理出现 bug，可能导致缓存污染或使用过期缓存，产生静默的错误输出。然而，当前实现通过组合 (num_rep, sp_size, mask shape, replay_token) 作为缓存键，设计相对健壮。
 3. 测试覆盖：新增的单元测试有效模拟了集合通信，并断言了与参考实现的数值接近。但测试环境 (mock) 与真实多 GPU 环境仍有差异，集成测试（如端到端的 Qwen-Image 测试）对于验证在真实硬件和集合通信中的行为至关重要。CI 状态显示部分测试套件失败 (Extra, AMD ROCm 7.2)，需要关注是否与本 PR 相关。
 4. 向后兼容性：该 PR 为现有功能添加了支持，未改变已有合法调用的行为。不合法组合（如 replicated suffix 或 ring 并行）仍会抛出 NotImplementedError，符合预期。
 - 影响：对用户的影响：直接解决了特定用户（使用多 GPU + SP + BCG 运行 Qwen-Image 等扩散模型）的完全阻塞性问题。修复后，用户可以成功运行 BCG 以获得推理加速，而无需禁用 SP。性能数据显示，在 SP=2 下 BCG 重放仍能带来约 7.3% 的去噪时间节省。对系统的影响：增强了 sglang.multimodal_gen 中 USPAttention 在复

杂并行场景下的鲁棒性和功能完整性。记忆化设计为 BCG 这种需要高效重放的模式提供了基础设施支持。变更是局部的，主要影响 multimodal_gen 的注意力层，未波及其他后端或核心调度逻辑。对团队的影响：展示了处理复杂并行注意力中掩码和数据布局问题的方法论。修复了现有测试套件在主分支上的失败，提升了代码库的健康度。PR 的详细测试数据（准确性、性能）为类似优化提供了范例。

- 风险标记：核心路径变更，分布式通信逻辑，测试依赖 Mock

关联脉络

- PR #37674 [misc] Extract PP dynamic chunk sizing into a DynamicChunkSizer scheduler component: 同属序列并行 (SP) 和调度器组件优化，虽然模块不同（调度 vs 注意力），但共享对动态分块、KV 缓存和并行模式管理的关注。
- PR #37329 Improve CUDA graph and speculative execution output handling: 同属对 CUDA Graph 和推测执行路径的改进，本 PR 修复 BCG 在 SP 下的问题，而 #37329 优化 DSpark 推测解码的输出处理，反映了仓库在图模式推理优化上的持续投入。