PR #23292 完整报告

sgl-project/sglang

[CP] 1/N: Support MLA Prefill Context Parallel

合并时间：2026-05-23 18:07

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23292

执行摘要

- 一句话：为 MLA 模型添加预填充上下文并行支持
- 推荐动作：值得精读。本 PR 展示了如何通过精心的模块化设计，以最小改动复用 NSA CP 的大量基础设施（通信器、KV 收集、zigzag 分割），体现了 sglang 注意力层抽象的良好扩展性。尤其关注 flashattention_backend.py 中 _mla_cp_attn 的封装方式以及 cp_utils.py 中路由守卫的层次设计，是工程与算法结合的典型案例。

功能与动机

来自 PR body：'Extend SGLang's prefill context parallelism (CP) to MLA-based models (DeepSeek V3 / R1, Kimi K2.5) on the fa3 attention backend, unlocking multi-GPU long-prefill throughput for MLA architectures.' 关联 Issue #21788 (Context Parallelism 路线图) 和 #22896 (MLA CP 功能请求) 驱动了此变更。

实现拆解

1. 路由守卫与配置校验：在 cp_utils.py 中新增 is_mla_prefill_cp_enabled() 和 mla_use_prefill_cp() 函数，并在 server_args.py 中拒绝非 FA3 后端使用 --enable-prefill-context-parallel，与 NSA CP 标志互斥，强制 EP size = TP size。
2. 核心注意力闭包：在 flashattention_backend.py 的 forward_extend 中重构 KV 缓存写入逻辑：MLACP 下直接通过 set_mla_kv_buffer 写入全序列 KV（无需 allgather）；新增 _mla_cp_attn 闭包，封装 FA3 的 flash_attn_with_kvcache 调用，支持 zigzag 分割后的局部注意力计算。同时通过 page_table 展宽处理 padding 带来的越界问题。
3. 模型门控调整：在 deepseek_v2.py、deepseek_nextn.py、forward_mla.py 中，将六处 CP 门控从仅 NSA 放宽为“NSA 或 MLA CP”，使 MLA CP 能复用 NSA CP 的通信器、KVCache 重建等基础设施；同时通过 is_mla_prefill_cp_enabled() 在 communicator.py、cuda_graph_runner.py 等处限制 MHA CP 不被拉入 NSACPLayerCommunicator。
4. 数值对等性测试：新增 test_mla_cp_fa3_parity.py（单进程、单层、预填充 paged KV cache，验证 rank-local CP 输出与全序列非 CP FA3 一致）和 test_cp_prefix_len_fa3_parity.py（验证带有 prefix 的 CP 元数据正确性）。
5. 端到端测试与 CI 迁移：新增 test_deepseek_v3_cp_single_node.py（tp=8, dp=2, attn-cp=4，GSM8k 准确率 >= 0.935），将 DSA CP 测试统一迁移到 test/registered/cp/ 目录，并更新 CI 注册为新的 stage 命名。

关键文件：

- test/registered/kernels/test_mla_cp_fa3_parity.py（模块 注意力测试；类别 test；类型 test-coverage；符号 _build_cache_and_q, _full_seq_attn, _cp_attn_for_rank, _mla_cp_attn）：MLA CP 的 kernel 级数值对等性测试，验证 rank-local CP 输出与全序列非 CP FA3 一致的精度，是核心准确度保障。
- python/sglang/srt/layers/attention/flashattention_backend.py（模块 注意力后端；类别 source；类型 core-logic；符号 _mla_cp_attn, forward_extend, init_forward_metadata）：核心变更文件：新增 _mla_cp_attn 闭包、重构 forward_extend 中 KV 缓存写入逻辑，以及 page_table 展宽处理 padding。
- test/registered/kernels/test_cp_prefix_len_fa3_parity.py（模块 注意力测试；类别 test；类型 test-coverage；符号 TestCPPrefixLenFA3Parity, _run_parity, _call_meta, test_cp2_prefix1_extend3）：验证带 prefix 的 FA3 CP 元数据正确性的单元测试，确保 prepare_context_parallel_metadata 输出与全序列 FA3 一致。
- test/registered/cp/test_deepseek_v3_cp_single_node.py（模块 模型测试；类别 test；类型 test-coverage；符号 TestDeepseekV3CPInSeqSplit, setUpClass, tearDownClass, test_a_gsm8k）：端到端 GS M8k 准确性测试，使用真实 DeepSeek-V3-0324 模型，tp=8 dp=2 attn-cp=4，确保 MLA CP 在生产配置下达到与非 CP 相同的准确率。
- python/sglang/srt/layers/utils/cp_utils.py（模块 CP 工具；类别 source；类型 dependency-wiring；符号 is_mla_prefill_cp_enabled, mla_use_prefill_cp, can_cp_split, prepare_context_parallel_metadata）：新增 is_mla_prefill_cp_enabled() 和 mla_use_prefill_cp() 函数，修改 can_cp_split() 和 prepare_context_parallel_metadata()，是路由守卫和元数据计算的核心。
- python/sglang/srt/server_args.py（模块 配置层；类别 source；类型 core-logic；符号 _handle_model_specific_adjustments）：添加后端校验、标志互斥和 EP/TP 强制约束，防止用户错误配置。
- python/sglang/srt/models/deepseek_v2.py（模块 模型实现；类别 source；类型 data-contract；符号 DeepseekV2Attention.forward, DeepseekV2DecoderLayer.forward）：门控调整核心：将六处 NSA-only 条件拓宽为 NSA or MLA CP，使 MLA CP 复用 NSA CP 通信器和 KV 重建。同时修复多模态 input_embeds 为 None 的问题。

关键符号：_mla_cp_attn, is_mla_prefill_cp_enabled, mla_use_prefill_cp, cp_attn_forward_extend, prepare_context_parallel_metadata, can_cp_split

关键源码片段

test/registered/kernels/test_mla_cp_fa3_parity.py

MLA CP 的 kernel 级数值对等性测试，验证 rank-local CP 输出与全序列非 CP FA3 一致的精度，是核心准确度保障。

```
"""
FA3 numerical parity for MLA prefill CP.

Verifies the rank-local zigzag-split FA3 path (``_mla_cp_attn`` +
```

```
``cp_attn_forward_extend`` in ``flashattention_backend.py``) matches a
single non-CP ``flash_attn_with_kvcache`` over the full sequence.
"""


def _cp_attn_for_rank(rank, cp_size, block_size, q_nope, q_rope,
                      c_kv_cache, k_rope_cache, page_table, softmax_scale):
    """
    模拟 rank 本地 CP 路径：对 zigzag 分割的两块分别调用 FA3 MLA 注意力，
    然后拼接结果。这里直接调用 FA3 的 flash_attn_with_kvcache 以隔离后端逻辑。
    """
    num_blocks = cp_size * 2
    b_prev, b_next = rank, num_blocks - 1 - rank
    prev_slice = slice(b_prev * block_size, (b_prev + 1) * block_size)
    next_slice = slice(b_next * block_size, (b_next + 1) * block_size)

    # 取当前 rank 负责的两块 query 并融合 nope+rope
    q_nope_local = torch.cat([q_nope[prev_slice], q_nope[next_slice]], dim=0)
    q_rope_local = torch.cat([q_rope[prev_slice], q_rope[next_slice]], dim=0)
    q_fused = torch.cat([q_nope_local, q_rope_local], dim=-1)

    # 构造 ContextParallelMetadata 模拟生产环境
    cp_meta = ContextParallelMetadata(
        kv_len_prev_tensor=torch.tensor([(b_prev + 1) * block_size], dtype=torch.int32, device=
        DEVICE),
        kv_len_next_tensor=torch.tensor([(b_next + 1) * block_size], dtype=torch.int32, device=
        DEVICE),
        actual_seq_q_prev=block_size,
        actual_seq_q_next=block_size,
    )
    fb = SimpleNamespace(attn_cp_metadata=cp_meta)

    def _mla_cp_attn(q_chunk, cu_seqlens_q_cp, cache_seqlens_cp, max_seqlen_q_cp):
        """
        对 rank 本地的 query 块调用 FA3 的 MLA 吸收注意力。
        注意：FA3 要求 q 和 qv 分开传入，这里从融合的 q_chunk 中拆开。
        """
        q_nope_chunk = q_chunk[..., :V_HEAD_DIM]
        q_rope_chunk = q_chunk[..., V_HEAD_DIM:]
        return flash_attn_with_kvcache(
            q=q_rope_chunk, qv=q_nope_chunk,
            k_cache=k_rope_cache, v_cache=c_kv_cache,
            page_table=page_table,
            cache_seqlens=cache_seqlens_cp,
            cu_seqlens_q=cu_seqlens_q_cp,
            softmax_scale=softmax_scale, causal=True, ver=3,
        )

    # 调用生产路径的 cp_attn_forward_extend 以复用相同调度逻辑
    return cp_attn_forward_extend(q_fused, cp_meta, _mla_cp_attn, fb)
```

python/sglang/srt/layers/attention/flashattention_backend.py

核心变更文件：新增 _mla_cp_attn 闭包、重构 forward_extend 中 KV 缓存写入逻辑，以及 page_table 展宽处理 padding。

```
# flashattention_backend.py 中 forward_extend 的关键修改片段

def forward_extend(self, q, k, v, layer, forward_batch, save_kv_cache=True,
                   q_rope=None, k_rope=None, sinks=None):
    is_cp_mode = (
        forward_batch.forward_mode.is_context_parallel_extend()
        and forward_batch.attn_cp_metadata is not None
        and self.attn_cp_size > 1
    )

    if k is not None:
        # 注：MLACP 下 k 和 k_rope 已是全序列（rebuild_cp_kv_cache 已执行），
        # 因此直接写入 rank 的 pool 即可，无需 allgather
        if save_kv_cache and not self.fa_skip_kv_cache:
            cache_loc = forward_batch.out_cache_loc # 未 zigzag 分割，写入正确位置
            if self.use_mla:
                # MLA CP: 全序列 KV 写入，out_cache_loc 在所有 rank 上一致
                self.token_to_kv_pool.set_mla_kv_buffer(layer, cache_loc, k, k_rope)
            elif is_cp_mode:
                # 非 MLA 但 CP 模式：仍需要 allgather
                cp_allgather_and_save_kv_cache(forward_batch, layer, k, v, self.attn_cp_size)
            else:
                self.token_to_kv_pool.set_kv_buffer(layer, cache_loc, k, v,
                                                    layer.k_scale, layer.v_scale)

    # 在 init_forward_metadata 中添加 page_table 展宽：
    # 由于 prepare_mlp_sync_batch 会将 extend tokens 填充到 cp_size 的倍数，
    # cache_seqlens_cp 可能超过实际 seq_len，需要展宽 page_table 以保证 FA3
    的因果读取不越界。
    # 展宽的列指向 KV slot 0（req_to_token 零初始化），对应 padding query
    的输出会在下游丢弃。
    if (self.attn_cp_size > 1
        and forward_batch.global_num_tokens_cpu is not None
        and forward_batch.extend_num_tokens is not None
        and forward_batch.extend_seq_lens_cpu is not None):
        padded_extend = int(forward_batch.extend_num_tokens)
        real_extend = int(sum(forward_batch.extend_seq_lens_cpu))
        pad_delta = padded_extend - real_extend
        if pad_delta > 0:
            metadata.max_seq_len_k += pad_delta
```

评论区精华

- 测试目录统一：Fridge003 指出应将 test_qwen3_30b.py、test_deepseek_v32_cp_single_node.py 以及 TestDSV4FlashFP4B200Balanced_CP 迁移到 test/registered/cp/ 目录，使 CP 测试有统一归属。作者随后执行了该改动。
- 硬编码 disable_pcg：Fridge003 质疑 server_args.py 中在 DSA CP 下硬编码 disable_pcg=True 的原因，作者承认是遗留代码，需重新审查，后已移除。
- 全局缓存配置值：Fridge003 建议将 mla_enable_prefill_cp 和 dsa_enable_prefill_cp 的计算结果全局存储，避免重复调用。作者确认并统一修改。
- input_embeds 为 None 的兼容：whybeyoung 指出 deepseek_v2.py 中 forward 方法在 input_ids 为 None 时（多模态模型）应检查 input_embeds 的 shape。作者修复了该问题。
- CP 测试目录统一 (testing): 作者执行了迁移，并在后续提交中完成了目录调整和文件移动。
- 硬编码 disable_pcg 的原因 (design): 作者后续移除了该硬编码（在最终合并的版本中已删除）。
- 全局缓存 mla_enable_prefill_cp 配置值 (performance): 作者同意并统一修改，在 DeepseekV2Attention 的 __init__中缓存该值。
- input_embeds 为 None 的兼容性 (correctness): 作者添加了 if input_ids is not None 的判断，回退到 input_embeds.shape[0]。

风险与影响

- 风险：
  1. 后端绑定风险：MLA CP 仅支持 FA3 attention 后端（sm90+），若用户使用其他后端会直接报错并退出，属于硬约束，需在文档中强调。
  2. batch_size 限制：当前 CP 实现仅支持 batch_size=1，can_cp_split 中有显式断言，对于多 batch 请求会静默跳过 CP，性能可能低于预期。
  3. 强耦合 EP 配置：server_args.py 中强制 EP size = TP size 且禁用 pipeline parallelism，可能与部分部署需求冲突，需后续 PR 解耦。
  4. 数值精度边界：kernel 测试校准在 atol=5e-3 / rtol=1e-3，但对于非常长的序列可能因累加次序引入更大差异，生产环境需监控。
  5. 多模态模型兼容性：仅处理了 input_embeds 为 None 的情况，后续可能有更多模态数据路径需适配。 - 影响：对使用 DeepSeek V3/R1、Kimi K2.5 等 MLA 模型的用户，现在可通过 --enable-prefill-context-parallel --attention-backend fa3 启用预填充上下文并行，在长 prompt 场景下获得多 GPU 线性加速。对现有 MHA/NSA CP 用户无影响（通过条件守卫隔离）。系统层面增加了 FA3 后端的强依赖，仅 Hopper 及以上 GPU 可用。团队维护了统一的 CP 测试目录，为后续 decode CP 和跨后端支持奠定基础。 - 风险标记：仅 FA3 后端，batch_size=1 限制，EP size=TP size 硬约束，多模态兼容性待验证

关联脉络

- PR #21788 [Roadmap] Context Parallelism (2026 Q2): 此 PR 是路线图中的“Prefill CP for MLA models”子项，直接对应 roadmap 任务。

- PR #22896 [Feature] Prefill Context Parallelism Support for MLA Models: 该 issue 为本 PR 的功能请求，PR 关闭后即可 close #22896。
- PR #22692 [Bug] KIMI-K2.5 can't use context parallel: 本 PR 通过支持 MLA CP 修复了 Kimi K2.5 无法使用 context parallel 的问题，PR body 中声明合并后将关闭此 bug。
- PR #18233 Prefill CP for MHA/GQA model (Qwen3-MoE + FA3): MHA CP 的先驱 PR，本 PR 在其基础上扩展了 MLA 支持，并复用了类似的分割和通信模式。