PR #24947 完整报告

sgl-project/sglang

DeepSeek V4: Support context parallelism with fused MoE (non-DeepEP)

合并时间：2026-06-02 05:25

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24947

执行摘要

- 一句话：DeepSeek V4 CP 支持 Fused MoE，降低内存开销
- 推荐动作：建议精读本 PR，尤其关注 CP 如何与 MoE 后端解耦的设计。虽然 reviewer 建议的通信抽象未完全落地，但当前的实现已经解决了 H20-3e 部署的核心问题。对于其他 GPU 平台，需要补充对应的 Triton 配置文件。

功能与动机

目前 DeepSeek-V4 的上下文并行（CP）在 H20-3e 部署中只能与 DeepEP 配合使用，但 TP8 Fused MoE 在 H20-3e 上比 DeepEP 性能优势明显，且 DeepEP CUDA graph 消耗约 20GB 内存，严重限制可用内存和 batch size。DeepSeek-V4 Pro 在 TP16+PP2 配置下也无法支撑实际工作负载。需要解除这一限制，使 CP 能够与 Fused MoE、Marlin MoE 等后端协作。

实现拆解

1. 通信函数抽象（communicator_dsa_cp.py）：新增 dsa_cp_gather_hidden_states 和 dsa_cp_reduce_scatter_hidden_states 函数，用于在 CP 下对 hidden states 进行 all-gather 和 reduce-scatter。同时重构类内方法消除重复代码。
2. input_ids 重排（cp_utils.py）：新增 cp_round_robin_input_ids 函数，在 round-robin split 模式下，根据 MoE 后端类型对 input_ids 进行不同布局：non-DeepEP 时产生完整 round-robin 序列，DeepEP 时切分 per-rank。
3. 模型前向调整（deepseek_v4.py）：在 DeepseekV4DecoderLayer.forward 和 DeepseekV4Model.forward 中，当 CP 启用且 MoE 后端为 none 时，使用新通信函数实现 hidden_states 的 gather/reduce-scatter，并设置 use_reduce_scatter=True 跳过 MLP 内部 all-reduce。input_ids 在 model.forward 中调用 cp_round_robin_input_ids 对齐布局。
4. Triton 调优配置：新增 6 个 JSON 文件，针对 FP8 w8a8、block_shape [128,128]、不同 MoE 规模的 fused MoE runner 在 H20/H20-3e 上的 forward 和 forward-down 配置。
5. 测试验证：添加 TestDSV4FlashFP4B200Balanced_CP_NonDeepEP 测试类，在 TP4+CP4+EAGLE 配置下运行基本功能和精度测试，所有 case 通过。

关键文件：

- python/sglang/srt/layers/communicator_dsa_cp.py（模块 通信层；类别 source；类型 core-logic；符号 dsa_cp_gather_hidden_states, dsa_cp_reduce_scatter_hidden_states）：核心通信函数实现，新增 dsa_cp_gather_hidden_states 和 dsa_cp_reduce_scatter_hidden_states 用于 CP 下 hidden states 的 gather/reduce-scatter，并重构内部方法消除冗余。
- python/sglang/srt/layers/utils/cp_utils.py（模块 CP 工具；类别 source；类型 core-logic；符号 cp_round_robin_input_ids）：新增 cp_round_robin_input_ids 函数，根据 MoE 后端对 input_ids 进行不同的 round-robin 重排，确保输入布局与 MoE 处理一致。
- python/sglang/srt/models/deepseek_v4.py（模块 模型前向；类别 source；类型 data-contract）：模型前向修改：在 DeepseekV4DecoderLayer.forward 中为 CP+non-DeepEP 添加 hidden_states 的 gather/reduce-scatter，在 DeepseekV4Model.forward 中添加 input_ids 的 round-robin 重排。
- test/registered/cp/test_deepseek_v4_flash_fp4_b200_cp.py（模块 测试验证；类别 test；类型 test-coverage；符号 TestDSV4FlashFP4B200Balanced_CP_NonDeepEP, setUpClass, tearDownClass）：新增测试类 TestDSV4FlashFP4B200Balanced_CP_NonDeepEP，验证 CP+non-DeepEP（fused MoE）在 FP4+EAGLE 配置下的正确性和精度。
- python/sglang/srt/layers/moe/moe_runner/triton_utils/configs/triton_3_5_1/E=256,N=256,device_name=NVIDIA_H20-3e,dtype=fp8_w8a8,block_shape=[128, 128]_down.json（模块 配置调优；类别 config；类型 configuration）：新增 Triton kernel 调优配置，针对 H20-3e 的 fused MoE runner，覆盖 E=256,N=256 的 forward-down 方向。提升 fused MoE 在 H20-3e 上的性能。

关键符号：dsa_cp_gather_hidden_states, dsa_cp_reduce_scatter_hidden_states, cp_round_robin_input_ids

关键源码片段

python/sglang/srt/layers/communicator_dsa_cp.py

核心通信函数实现，新增 dsa_cp_gather_hidden_states 和 dsa_cp_reduce_scatter_hidden_states 用于 CP 下 hidden states 的 gather/reduce-scatter，并重构内部方法消除冗余。

```
def dsa_cp_gather_hidden_states(hidden_states: torch.Tensor):
    # 在 CP 下 all-gather hidden states，使后续 MoE 处理完整序列
    # 当前仅支持 attn_dp_size=1 和 attn_tp_size=1（round-robin split 的限制）
    attn_dp_size = get_attention_dp_size()
    attn_tp_size = get_attention_tp_size()
    assert attn_dp_size == 1 and attn_tp_size == 1
    # 获取本地 DP buffer 作为输出，当前 rank 的 hidden_states 作为输入
    hidden_states, local_hidden_states = (
        get_local_dp_buffer(get_attention_cp_group()),
        hidden_states,
    )
    # 所有 CP rank 都贡献其 local_hidden_states，得到完整序列
    attn_cp_all_gather_into_tensor(hidden_states, local_hidden_states)
```

```
    return hidden_states


def dsa_cp_reduce_scatter_hidden_states(hidden_states: torch.Tensor):
    # 在 MoE 后 reduce-scatter hidden states，恢复 per-rank 切分
    attn_dp_size = get_attention_dp_size()
    attn_tp_size = get_attention_tp_size()
    assert attn_dp_size == 1 and attn_tp_size == 1
    cp_size = get_attention_cp_size()
    cp_rank = get_attention_cp_rank()
    # 保存输入作为 reduce-scatter 的源 buffer
    input_hidden_states = hidden_states
    # 按 CP size 分割，提取当前 rank 的对应块
    hidden_states = hidden_states.tensor_split(cp_size)[cp_rank]
    # 执行 reduce-scatter：各 rank 求和后分发
    attn_cp_reduce_scatter_tensor(hidden_states, input_hidden_states)
    return hidden_states
```

python/sglang/srt/layers/utils/cp_utils.py

新增 cp_round_robin_input_ids 函数，根据 MoE 后端对 input_ids 进行不同的 round-robin 重排，确保输入布局与 MoE 处理一致。

```
def cp_round_robin_input_ids(input_ids):
    # CP round-robin split 模式下对 input_ids 进行重排
    # 输入：每个 rank 持有完整序列的一部分（如 rank 0~7 持有 0,1,2,3,4,5,...）
    # non-DeepEP：所有 rank 获得完整序列但按 round-robin 重排（0,8,16,...,1,9,17,...）
    # DeepEP：每个 rank 只持有对应切分的子序列（rank 0: 0,8,16,...）
    cp_size = get_attention_cp_size()
    cp_rank = get_attention_cp_rank()
    if get_moe_a2a_backend().is_none():
        # non-DeepEP：reshape 并转置展平
        input_ids = input_ids.reshape(-1, cp_size).T.flatten()
    else:
        # DeepEP：直接按 stride 切分
        input_ids = input_ids[cp_rank::cp_size].contiguous()
    return input_ids
```

评论区精华

- use_reduce_scatter 参数是否有用：xu-yfei 解释在 fused MoE 中需要设置 use_reduce_scatter=True 来跳过 MLP 内部的 all-reduce，配合外部的 reduce-scatter 完成正确通信。
- assert attn_tp_size == 1 的限制：xu-yfei 补充说明当前 round-robin split 模式仅支持 attn_dp_size=1 和 attn_tp_size=1，这是 server_args 中的强制约束，未来可扩展。
- 通信操作理想位置：Fridge003 建议将通信调用封装到 communicator_dsa_cp.py 中，xu-yfei 认为直接集成需要对 MLP 和 attention 进行大规模重构，留待后续。

- cp_round_robin_input_ids 与现有切分的差异：xu-yfei 说明 non-DeepEP 下 MoE 输入是完整 round-robin 序列，DeepEP 下是 per-rank 切分，两者不同，因此需要独立函数处理 non-DeepEP 情况。
  - use_reduce_scatter 参数是否必要 (design): 参数是必要的，用于控制 MoE 内部是否执行 all-reduce，已保留。
  - assert attn_tp_size == 1 是否影响其他模型 (design): 当前限制合理，未来可通过扩展 server_args 支持更通用场景。
  - 通信操作应封装在 communicator_dsa_cp.py 中 (design): 当前实现接受在模型文件中直接调用，未来可重构时优化。
  - cp_round_robin_input_ids 与现有切分逻辑是否重复 (correctness): 两种路径的 input_ids 布局不同，独立函数是合理的。

风险与影响

- 风险：
  - 核心路径变更：deepseek_v4.py 的 forward 流程修改，CP 启用时的条件分支（_use_cp、_use_tp_moe_gather、_use_tp_attn_a2a_scatter）可能在其他配置下引入回归，需完整回归测试。
  - 通信正确性：新增的 gather/reduce_scatter 在 CP 组内操作，假设 attn_dp_size=attn_tp_size=1，若未来扩展需验证其他配置下的行为。
  - 硬件特定配置：Triton kernel 配置文件仅针对 H20/H20-3e 添加，其他 GPU（如 H100、B200）可能缺少对应调优，导致性能退化或 fallback 到默认配置。
  - 测试覆盖有限：测试仅在 FP4+EAGLE 单一配置下运行，标准 FP8 或纯 CP 无 spec 的场景未覆盖，可能遗漏兼容性问题。
- 影响：
  - 对用户：DeepSeek-V4 在 H20-3e 上部署时可选择 Fused MoE（--moe-a2a-backend none），节省约 20GB CUDA graph 内存，提升可用 KV cache 和 batch size，同时获得更好的推理性能。
  - 对系统：不影响现有 DeepEP CP 路径，用户只需移除 --moe-a2a-backend deepep 即可自动使用新能力。
  - 对团队：新增的通信函数为后续其他模型（如 DS V2、DS V3）的 CP 扩展提供了可复用基础。
  - 风险标记：核心路径变更，硬件特定配置，通信正确性，测试覆盖有限

关联脉络

- PR #24704 Some PR for CP: PR body 中提到与本 PR 结合进行 CP8PP2 测试，可能涉及相同的 CP 基础设施。