PR #28102 完整报告

sgl-project/sglang

Fix DP attention + EP mode of Nemotron

合并时间：2026-06-13 09:39

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/28102

执行摘要

- 一句话：修复 Nemotron DP attention + EP 模式下多冗余 AllReduce
- 推荐动作：该 PR 值得精读，尤其是对实现 DP attention 和 EP 混合并行的工程师。核心设计决策是：在 MoE 场景下，expert 计算后已通过 EP 隐式通信，后续只需一次 ReduceScatter 即可完成聚合，不应再额外做 AllReduce。通过 skip_all_reduce 和 should_skip_post_experts_all_reduce 的组合，优雅地解决了这一冗余通信问题。

功能与动机

通过 profiling 发现，在同时启用 DP attention 和 EP 时，Nemotron 模型的前向计算中出现了冗余的 AllReduce。PR body 中明确说明"It will perform AR + RS, which the only necessary comm op after the experts should be RS here."，即本应只有一次 ReduceScatter，但代码错误地额外做了 AllReduce，导致通信开销翻倍，严重影响性能。

实现拆解

1. 引入 should_skip_post_experts_all_reduce 工具函数：在 nemotron_h.py 中，从 sglang.srt.layers.moe.utils 新增导入 should_skip_post_experts_all_reduce。该函数根据是否启用 TP 路径和是否允许 ReduceScatter 来决定是否跳过 AllReduce。
2. 修改 MLP 层 forward 方法：为 NemotronHMLP.forward 添加 use_reduce_scatter 参数，当该参数为 True 时，调用 self.down_proj(x, skip_all_reduce=True) 跳过内部的 AllReduce，保留张量为分片状态，以配合后续的 ReduceScatter。
3. 修改 MoE 层 forward 方法：为 NemotronHMoE.forward 添加 use_reduce_scatter 参数，在 MoE 计算完成后，原本 if self.tp_size > 1: 的 AllReduce 条件改为同时检查 should_skip_post_experts_all_reduce(is_tp_path=True, use_reduce_scatter=use_reduce_scatter)，当允许 ReduceScatter 时跳过此 AllReduce。
4. 在 MLP-like 解码层中传递 ReduceScatter 标志：修改 NemotronHMLPLikeDecoderLayer.forward，通过 self.layer_communicator.should_use_reduce_scatter(forward_batch) 获取当前批是否应使用 ReduceScatter，并将其传给 self.mixer.forward。
5. 配置 allow_reduce_scatter：在 NemotronHMLPLikeDecoderLayer 和 NemotronHMoEDecoderLayer 的初始化中，调用 make_layer_communicator(self.norm, for_attn=False, allow_reduce_scatter=True)，启用 ReduceScatter 能力。

6. 更新 nemotron_h_utils.py：在 make_layer_communicator 函数中添加 allow_reduce_scatter 参数（默认为 False），并通过 LayerCommunicator 的构造函数传递，控制通信模式。

关键文件：

- python/sglang/srt/models/nemotron_h.py（模块 模型实现；类别 source；类型 core-logic；符号 forward）：核心模型文件，修改了 MLP、MoE 和 MLP-like decoder layer 的 forward 方法，实现跳过冗余 AllReduce 的核心逻辑。
- python/sglang/srt/models/nemotron_h_utils.py（模块 模型工具；类别 source；类型 data-contract；符号 make_layer_communicator）：修改了 make_layer_communicator 函数，新增 allow_reduce_scatter 参数并传递给 LayerCommunicator，是启用 ReduceScatter 的配置入口。

关键符号：forward, make_layer_communicator

关键源码片段

python/sglang/srt/models/nemotron_h.py

核心模型文件，修改了 MLP、MoE 和 MLP-like decoder layer 的 forward 方法，实现跳过冗余 AllReduce 的核心逻辑。

```
# NemotronHMLP: 支持通过 use_reduce_scatter 跳过 AllReduce
class NemotronHMLP(nn.Module):
    def forward(self, x: torch.Tensor, use_reduce_scatter: bool = False):
        x, _ = self.up_proj(x)
        x = self.act_fn(x)
        # 当 use_reduce_scatter 为 True 时，跳过 down_proj 内部的 AllReduce
        x, _ = self.down_proj(x, skip_all_reduce=use_reduce_scatter)
        return x


# NemotronHMoE: 根据 use_reduce_scatter 决定是否执行 post-experts AllReduce
class NemotronHMoE(nn.Module):
    def forward(
        self, hidden_states: torch.Tensor, use_reduce_scatter: bool = False
    ) -> torch.Tensor:
        num_tokens, hidden_dim = hidden_states.shape
        # ... 省略 MoE 计算过程 ...
        final_hidden_states, shared_output = ...
        # 合并 shared experts 输出
        if shared_output is not None:
            final_hidden_states += shared_output
        # 仅在允许 ReduceScatter 时跳过 AllReduce
        if self.tp_size > 1 and not should_skip_post_experts_all_reduce(
            is_tp_path=True,
            use_reduce_scatter=use_reduce_scatter,
        ):
            final_hidden_states = tensor_model_parallel_all_reduce(final_hidden_states)
```

```
        return final_hidden_states.view(num_tokens, hidden_dim)


# NemotronHMLPLikeDecoderLayer: 在 forward 中获取 use_reduce_scatter 并传递给 mixer
class NemotronHMLPLikeDecoderLayer(nn.Module):
    def forward(
        self,
        hidden_states: torch.Tensor,
        residual: torch.Tensor,
        forward_batch: ForwardBatch,
    ) -> tuple[torch.Tensor, torch.Tensor]:
        hidden_states, residual = self.layer_communicator.prepare_mlp(
            hidden_states, residual, forward_batch
        )
        # 通过 layer_communicator 判断当前批次是否应使用 ReduceScatter
        use_reduce_scatter = self.layer_communicator.should_use_reduce_scatter(
            forward_batch
        )
        hidden_states = self.mixer.forward(
            hidden_states, use_reduce_scatter=use_reduce_scatter
        )
        hidden_states, residual = self.layer_communicator.postprocess_layer(
            hidden_states, residual, forward_batch
        )
        return hidden_states, residual
```

python/sglang/srt/models/nemotron_h_utils.py

修改了 make_layer_communicator 函数，新增 allow_reduce_scatter 参数并传递给 LayerCommunicator，是启用 ReduceScatter 的配置入口。

```
# nemotron_h_utils.py: 允许在创建层的通信器时启用 ReduceScatter
def make_layer_communicator(
    layer_norm: RMSNorm, *, for_attn: bool, allow_reduce_scatter: bool = False
) -> LayerCommunicator:
    return LayerCommunicator(
        layer_scatter_modes=_build_layer_scatter_modes(),
        input_layernorm=layer_norm if for_attn else nn.Identity(),
        post_attention_layernorm=nn.Identity() if for_attn else layer_norm,
        force_layernorm_before_dp_gather=True,
        allow_reduce_scatter=allow_reduce_scatter, # 新增参数，控制是否允许 ReduceScatter
    )
```

评论区精华

该 PR 无 review 评论，仅有一位 reviewer（Zhichenzzz）批准并留言“LGTM! Thanks”，说明变更得到了充分认可。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 回归风险：变更集中在 Nemotron 模型的前向路径，且涉及通信模式切换。默认 use_reduce_scatter 为 False，因此对于未启用 DP attention 或 EP 的场景，行为不变。对于同时启用 DP attention 和 EP 的场景，如果 should_use_reduce_scatter 判断有误，可能导致张量未正确聚合，影响模型精度。PR body 中的精度测试显示 accuracy 从 0.000 提升至 0.983，说明修复前精度完全错误，修复后恢复正常。
  2. 性能风险：通过跳过多余的 AllReduce，理论上降低通信量，加速推理。测试显示 latency 从 109.131s 降至 71.723s，提升约 34%。
  3. 兼容性风险：仅影响 Nemotron 系列模型的 DP attention + EP 模式，不影响其他模型或配置。 - 影响：影响范围：仅针对 Nemotron 模型（如 Nemotron-3-Ultra-550B-A55B-NVFP4）在同时启用 DP attention（ --enable-dp-attention）和 EP（--ep-size）时的推理路径。其他模型或配置无影响。性能提升：在 8xDP + 8xEP 配置下，GSM8K 测试的 latency 降低约 34%，output throughput 略有下降（因修复后输出正确结果，token 数不同）。精度修复：由于修复前错误地多做了 AllReduce，导致输出完全错误（accuracy 0.000），修复后 accuracy 达到 98.3%。 - 风险标记：核心路径变更，通信模式切换

关联脉络

- PR #27720 [DeepSeek V3] Defer moe finalize and fused it with main stream add: 同为 MoE 模块的通信优化，涉及 MoE finalize 和 fused 操作，与本 PR 的 ReduceScatter 优化具有相似的技术领域。
- PR #27945 fix(moe): make FlashInfer A2A robust to collapsed global_num_tokens (moe_dense_tp_size NaN): 同为 MoE 相关的 bugfix，改动涉及 MoE token dispatcher 和通信鲁棒性。