PR #47006 完整报告

vllm-project/vllm

[Perf][Qwen] Replace MOE all-reduce with reduce-scatter

合并时间：2026-07-12 14:14

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/47006

执行摘要

- 一句话：将 MoE 通信从 all-reduce 替换为 reduce-scatter
- 推荐动作：建议合并（已合并）。该 PR 是一个典型的通信优化案例，值得关注其如何通过调整通信算子的时机和粒度来减少冗余数据移动。review 讨论中关于 padded reduce-scatter 的权衡也值得学习。

功能与动机

在 MoE 模型启用序列并行时，注意力层的 all-reduce 与 MoE 内的序列分片存在冗余通信。将 all-reduce 拆分为 reduce-scatter 与 all-gather 后，可合并为一次 reduce-scatter，降低通信开销。PR 作者在 issue 评论中说明："Previously, after attention, all_reduce was performed in o_proj, and then the model immediately entered MoE. Since MoE sequence parallelism is enabled by default along with TP, the tensor would then be chunked along the sequence dimension. We can directly optimize all_reduce (= reduce_scatter + all_gather) + split chunk into a single reduce_scatter(dim=sequence)."

实现拆解

1. 接口扩展：在 Qwen3NextAttention.__init__、Qwen3NextDecoderLayer.__init__、QwenGatedDeltaNetAttention.__init__ 中新增 reduce_results 参数（默认 True），并透传给下游的 RowParallelLinear 层，控制注意力输出投影是否执行 all-reduce。
2. 条件判断：在 Qwen3NextDecoderLayer.__init__ 和 Qwen3_5DecoderLayer.__init__ 中，根据 parallel_config.use_sequence_parallel_moe、pipeline_parallel_size == 1 以及层类型（是否为 MoE 层）计算 use_attn_reduce_scatter_for_moe 标志。当该标志为真时，将 reduce_results=False 传递给注意力层，使其跳过 all-reduce。
3. 通信替换：在 Qwen3NextSparseMoeBlock.forward 中，当 use_attn_reduce_scatter_for_moe 为真时，在注意力前向传播之后插入 tensor_model_parallel_padded_reduce_scatter 操作（依赖于 PR #47070 提供的 padded 版本），将注意力输出沿序列维度直接 reduce-scatter，而不是先 all-reduce 再分片。
4. 配套调整：在 Qwen3_5DecoderLayer 中同步相同逻辑；在 QwenGatedDeltaNetAttention 中接收并传递 reduce_results 参数到 out_proj。

关键文件：

- vllm/model_executor/models/qwen3_next.py（模块 模型层；类别 source；类型 core-logic；符号 forward, _all_gather_hidden_and_residual）：核心改动文件：修改了 Qwen3NextAttention 和 Qwen3NextDecoderLayer 的初始化接口，在 Qwen3NextSparseMoeBlock.forward 中插入 padded reduce-scatter 调用，并调整了序列并行切片与重组的条件。
- vllm/model_executor/models/qwen3_5.py（模块 模型层；类别 source；类型 data-contract）：与 qwen3_next.py 类似但逻辑更简单：直接根据 model_type 是否为 qwen3_5_moe_text 判断 MoE 层，并传递 reduce_results。
- vllm/model_executor/layers/mamba/gdn/qwen_gdn_linear_attn.py（模块 模型层；类别 source；类型 data-contract）：改动较小：为 QwenGatedDeltaNetAttention 新增 reduce_results 参数并透传给 out_proj 的 RowParallelLinear。

关键符号：Qwen3NextAttention.init, Qwen3NextSparseMoeBlock.forward, Qwen3NextDecoderLayer.init, QwenGatedDeltaNetAttention.init, Qwen3_5DecoderLayer.init

关键源码片段

vllm/model_executor/models/qwen3_next.py

核心改动文件：修改了 Qwen3NextAttention 和 Qwen3NextDecoderLayer 的初始化接口，在 Qwen3NextSparseMoeBlock.forward 中插入 padded reduce-scatter 调用，并调整了序列并行切片与重组的条件。

```
# vllm/model_executor/models/qwen3_next.py
# Qwen3NextDecoderLayer.__init__: 决定是否在注意力层跳过 all-reduce
parallel_config = vllm_config.parallel_config
config = vllm_config.model_config.hf_text_config
# 获取仅 MLP 的层索引列表
mlp_only_layers = config.mlp_only_layers if hasattr(config, "mlp_only_layers") else []
# 判断当前层是否为 MoE 层（非 MLP-only 并且位置满足 decoder_sparse_step）
is_moe_layer = (self.layer_idx not in mlp_only_layers) and (
    config.num_experts > 0 and (self.layer_idx + 1) % config.decoder_sparse_step == 0
)
# 仅在启用序列并行 MoE、无 pipeline 并行且为 MoE 层时使用 reduce_scatter
self.use_attn_reduce_scatter_for_moe = (
    parallel_config.use_sequence_parallel_moe
    and parallel_config.pipeline_parallel_size == 1
    and is_moe_layer
)

# 将 reduce_results=False 传递给注意力模块
if self.layer_type == "linear_attention":
    self.linear_attn = QwenGatedDeltaNetAttention(
        config, vllm_config=vllm_config, prefix=f"{prefix}.linear_attn",
        gqa_interleaved_layout=True,
        reduce_results=not self.use_attn_reduce_scatter_for_moe,
    )
```

```
elif self.layer_type == "full_attention":
    self.self_attn = Qwen3NextAttention(
        config, model_config=model_config, cache_config=cache_config,
        quant_config=quant_config, prefix=f"{prefix}.self_attn",
        reduce_results=not self.use_attn_reduce_scatter_for_moe,
    )
```

vllm/model_executor/models/qwen3_5.py

与 qwen3_next.py 类似但逻辑更简单：直接根据 model_type 是否为 qwen3_5_moe_text 判断 MoE 层，并传递 reduce_results。

```
# vllm/model_executor/models/qwen3_5.py
# Qwen3_5DecoderLayer.__init__ 中新增逻辑
parallel_config = vllm_config.parallel_config
is_moe_layer = config.model_type == "qwen3_5_moe_text"
self.use_attn_reduce_scatter_for_moe = (
    parallel_config.use_sequence_parallel_moe
    and parallel_config.pipeline_parallel_size == 1
    and is_moe_layer
)

# 传递给线性注意力
if self.layer_type == "linear_attention":
    self.linear_attn = QwenGatedDeltaNetAttention(
        config=config, vllm_config=vllm_config,
        prefix=f"{prefix}.linear_attn", gqa_interleaved_layout=False,
        reduce_results=not self.use_attn_reduce_scatter_for_moe,
    )
# 传递给全注意力
elif self.layer_type == "full_attention":
    self.self_attn = Qwen3NextAttention(
        config, model_config=model_config, cache_config=cache_config,
        quant_config=quant_config, prefix=f"{prefix}.self_attn",
        reduce_results=not self.use_attn_reduce_scatter_for_moe,
    )
```

vllm/model_executor/layers/mamba/gdn/qwen_gdn_linear_attn.py

改动较小：为 QwenGatedDeltaNetAttention 新增 reduce_results 参数并透传给 out_proj 的 RowParallelLinear。

```
# vllm/model_executor/layers/mamba/gdn/qwen_gdn_linear_attn.py
# QwenGatedDeltaNetAttention.__init__
def __init__(
    self,
    config: Qwen3NextConfig,
    vllm_config: VllmConfig,
    prefix: str = "",
    gqa_interleaved_layout=False,
    reduce_results: bool = True, # 新增参数，默认为 True
```

```
) -> None:
    # ... 初始化 ...
    self.out_proj = RowParallelLinear(
        self.value_dim,
        self.hidden_size,
        bias=False,
        input_is_parallel=True,
        reduce_results=reduce_results, # 传递给线性层
        quant_config=self.quant_config,
        prefix=f"{prefix}.out_proj",
    )
```

评论区精华

- ZJY0516提问：“Is this a free lunch? Does it offer benefits across all scenarios when enabled?” 作者 gcanlin回复：“Theoretically, yes. It mainly reduces two parts of overhead:
  1. Previously, after attention, all_reduce was performed in o_proj, and then the model immediately entered MoE. ... 2. We do not perform all_gather immediately after MoE finishes.” 结论：该优化在理论上对所有场景有益，但实际收益可能因模型和配置而异。 - yewentao256在 review 评论中指出：“this path you might go through MTP, this might be an potential issue, you might also check them.” 作者回应已参考 PR #47070 采用 padding 方式解决，并更新了 PR 描述。
- 优化是否在所有场景下都免费受益 (performance): 作者确认理论上普遍受益，实际测试显示吞吐提升 2-3%。
- padded reduce-scatter 实现与 MTP 兼容性 (correctness): 作者已根据建议使用 padding 方法修复，未发现剩余问题。

风险与影响

- 风险：
  - 功能依赖：该 PR 依赖于 PR #47070 的 tensor_model_parallel_padded_reduce_scatter 操作，若 #47070 未合并则无法工作。
  - 场景限制：仅在 use_sequence_parallel_moe=True 且 pipeline_parallel_size==1 时生效，其他场景行为不变。
  - 潜在精度影响：由于通信顺序改变（all-reduce → reduce-scatter + 延迟的 all-gather），浮点累加顺序可能变化，导致微小精度差异。PR 提供的 GSM8K 评测结果显示精度持平（差异 <0.5%），风险可控。
  - MTP 兼容性：yewentao256 提醒在 MTP（多 token 预测）路径下可能存在问题，作者已通过 padding 方式规避，但缺乏直接测试覆盖。
- 影响：
  - 用户侧：Qwen3Next 和 Qwen3.5 MoE 模型用户在启用 --enable-expert-parallel（启用序列并行）时将自动获得 2-3% 的端到端吞吐提升（基于 PR 测试数据：总 token 吞吐从 20528→21013 tok/s），且无需调整其他配置。

- 系统侧：新增的 reduce_results 参数与现有通信模式兼容，非 MoE 层或 pipeline parallel >1 时降级为原始行为。
- 团队侧：代码改动集中在 3 个文件，设计清晰，但需确保与未来模型（如 DeepSeek 系列）的 MoE 实现保持一致性。
- 风险标记：依赖先行 PR#47070，仅 MoE 序列并行场景生效，MTP 路径潜在风险已规避

关联脉络

- PR #47070 Add tensor_model_parallel_padded_reduce_scatter operation: 本 PR 依赖 #47070 提供的 padded reduce-scatter 原语，否则无法实现非对齐序列长度的 reduce-scatter。
- PR #46635 Support sequence parallelism for MoE layers: 本 PR 是 #46635 的后续，在支持序列并行的基础上进一步优化通信模式。