PR #27826 完整报告

sgl-project/sglang

Optimize FLUX.1 tensor parallel sharding

合并时间：2026-06-12 13:13

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27826

执行摘要

- 一句话：优化 FLUX.1 的 TP 分片，2GPU 延迟降低 35%
- 推荐动作：PR 的核心设计决策（保留 Nunchaku 路径、权重分片加载）值得关注，适合对扩散模型 TP 优化感兴趣的工程师精读。mask 注意力 GQA 修复和 overlay 缓存完善也具有通用参考价值。

功能与动机

根据 PR 描述，之前的 FLUX.1 路径使用了许多 ColumnParallelLinear(..., gather_output=True) 投影和 full 注意力头，导致在 TP 下激活被提前收集回完整形状，使得多 GPU TP 不能有效减少 denoiser 计算量。FLUX.2 已经遵循分片模式，此项改动使 FLUX.1 更接近该执行模型。

实现拆解

1. 引入新依赖：在 flux.py 中导入 RowParallelLinear、divide 和 get_tp_world_size，为分片线性层和并行计算做准备。
2. 新增分片 MLP 模块：定义 FluxGELU 和 FluxParallelFeedForward 类，使用 ColumnParallelLinear(gather_output=False) 作为输入门控投影，RowParallelLinear(input_is_parallel=True) 作为输出投影，替代原来的 FeedForward 实现。
3. 修改注意力层：在 FluxAttention.__init__ 中根据 get_tp_world_size() 和 quant_config 判断是否分片（非 Nunchaku 且 tp_size>1 时启用 shard_qkv），将 QKV 投影的 gather_output 设为 not self.shard_qkv，并计算 local_heads = divide(heads, tp_size) 用于后续注意力计算。
4. 添加权重加载适配器：定义 _patch_proj_out_weight_loader 函数，将 proj_out 权重的加载逻辑改写为从 checkpoint 的 [attn_full | mlp_full] 拼接中提取对应分片，兼容 TP 分片后的参数分布。
5. 修复 mask 注意力 GQA 兼容性与缓存验证：在 layer.py 的 LocalAttention 中，当 q_.shape[1] != k_.shape[1] 时使用 repeat_interleave 分组扩展 k/v 头数。同时在 model_overlay.py 中增加 _materialized_overlay_cache_complete 验证函数，在 test_utils.py 中增加 Ascend 一致性 case 识别和 GT 图片加载重试逻辑，并在 test_transformer_quant.py 中添加 TP size 的 mock。

关键文件：

- python/sglang/multimodal_gen/runtime/models/dits/flux.py（模块 扩散模型；类别 source；类型 core-logic；符号 FluxGELU, init, forward, FluxParallelFeedForward）：核心变更文件，包含 FLUX.1 模型分片逻辑的所有新增和修改：FluxGELU、FluxParallelFeedForward、FluxAttention 分片改造以及权重加载适配器。
- python/sglang/multimodal_gen/runtime/utils/model_overlay.py（模块 模型叠加；类别 source；类型 data-contract；符号 _component_has_weight_file, _materialized_overlay_has_component_weights, _materialized_overlay_cache_complete）：新增 overlay 缓存完整性验证函数，避免材质化过程中的权重文件丢失问题，提高扩散模型加载可靠性。
- python/sglang/multimodal_gen/runtime/layers/attention/layer.py（模块 注意力层；类别 source；类型 core-logic）：修复 mask 注意力路径中 GQA 时 q 与 k/v 头数不匹配的问题，支持分组查询注意力，影响扩散模型双流注意力的正确性。
- python/sglang/multimodal_gen/test/test_utils.py（模块 测试工具；类别 test；类型 test-coverage；符号 _is_ascend_consistency_case, _load_remote_gt_image）：增加 Ascend 一致性 case 识别和 GT 图片加载重试逻辑，提升测试工具在特定硬件下的健壮性，与 overlay 缓存验证配合确保 CI 稳定性。
- python/sglang/multimodal_gen/test/unit/test_transformer_quant.py（模块 量化测试；类别 test；类型 test-coverage）：为单元测试添加 get_tp_world_size 的 mock，使 FLUX.1 分片逻辑的测试能在非 TP 环境下运行，保证测试有效性。

关键符号：FluxGELU.init, FluxGELU.forward, FluxParallelFeedForward.init, FluxParallelFeedForward.forward, FluxAttention.init(修改), _patch_proj_out_weight_loader, _materialized_overlay_cache_complete, _component_has_weight_file, _load_remote_gt_image, _is_ascend_consistency_case

关键源码片段

python/sglang/multimodal_gen/runtime/models/dits/flux.py

核心变更文件，包含 FLUX.1 模型分片逻辑的所有新增和修改：FluxGELU、FluxParallelFeedForward、FluxAttention 分片改造以及权重加载适配器。

```
class FluxParallelFeedForward(nn.Module):
    """
    FLUX.1 的分片 MLP 模块。与原生 FeedForward 不同，它使用 ColumnParallelLinear 作为
    第一个 GELU 门控投影（gather_output=False），并用 RowParallelLinear 作为输出投影
    （input_is_parallel=True），从而在 TP 下保持中间激活分片，减少通信量。
    """
    def __init__(
        self,
        dim: int,
        dim_out: Optional[int] = None,
        mult: int = 4,
        inner_dim: Optional[int] = None,
        bias: bool = True,
```

```
        quant_config: Optional[QuantizationConfig] = None,
        prefix: str = "",
    ):
        super().__init__()
        if inner_dim is None:
            inner_dim = int(dim * mult)
        dim_out = dim_out if dim_out is not None else dim

        # 分片门控线性层，gather_output=False 避免提前全收集
        self.net = nn.ModuleList([
            FluxGELU(
                dim,
                inner_dim,
                bias=bias,
                quant_config=quant_config,
                prefix=f"{prefix}.net.0" if prefix else "net.0",
            ),
            nn.Dropout(0.0),
            RowParallelLinear(
                inner_dim,
                dim_out,
                bias=bias,
                input_is_parallel=True, # 输入已经是分片的，不需要 all-gather
                quant_config=quant_config,
                prefix=f"{prefix}.net.2" if prefix else "net.2",
            ),
        ])

    def forward(self, hidden_states: torch.Tensor) -> torch.Tensor:
        hidden_states = self.net[0](hidden_states) # FluxGELU：分片线性 + gelu
        hidden_states = self.net[1](hidden_states) # Dropout
        hidden_states, _ = self.net[2](hidden_states) # RowParallelLinear：输出投影，内部处理 all-
        reduce
        return hidden_states

# FluxAttention.__init__ 中新增的分片条件判断（部分）
self.tp_size = get_tp_world_size()
# 仅在非 Nunchaku 量化且 TP 大于 1 时启用分片（Nunchaku 路径保持原 full-gather 行为）
self.shard_qkv = self.tp_size > 1 and not isinstance(
    quant_config, NunchakuConfig
)
self.local_heads = divide(self.heads, self.tp_size)

# QKV 投影根据 shard_qkv 开关决定 gather_output
# 原为 gather_output=True，现在改为动态
self.to_qkv = MergedColumnParallelLinear(
    query_dim,
    [self.inner_dim] * 3,
    bias=bias,
```

```
        gather_output=not self.shard_qkv, # 分片时停用全收集
        quant_config=quant_config,
        prefix=f"{prefix}.to_qkv" if prefix else "to_qkv",
    )
```

评论区精华

本 PR 由作者自行合并，未发现外部 review 讨论。作者在 commits 中曾尝试调整 CI 阈值后又回退，说明 CI 基线需要谨慎更新。

- 暂无高价值评论线程

风险与影响

- 风险： 1) 双路径维护风险：Nunchaku 量化路径保持原 full-gather 行为，与非 Nunchaku 分片路径形成两条代码路径，未来修改时需要考虑两者一致性。 2) weight loader 兼容性：新增的 _patch_proj_out_weight_loader 假设 checkpoint 中 proj_out 权重为 [attn_full | mlp_full] 拼接格式，若社区 checkpoint 格式不同，加载会失败。 3) mask 注意力内存消耗：在 layer.py 中添加的 repeat_interleave 在 q 头数远大于 k 头数时会显著增大 k/v 张量，可能导致高分辨率输入时显存压力上升。 4) overlay 缓存验证变慢：_materialized_overlay_cache_complete 遍历所有组件目录检查权重文件，可能使首次加载时间增加。
  - 影响：用户：多 GPU 部署 FLUX.1 的用户将获得显著的延迟改善（2GPU 约 35%），单 GPU 用户无影响。系统：代码库中扩散模型模块新增了两个 MLP 类、一个 weight loader 和一个分片条件判断，整体复杂度略有增加。团队：需要维护两条 TP 路径，但 weight loader 抽象降低了修改门槛。测试：配套测试增加了 Ascend 场景和重试逻辑，提高了 CI 稳定性。
  - 风险标记：双路径维护，weight loader 兼容性，mask 注意力内存增长

关联脉络

- 暂无明显关联 PR