PR #1848 完整报告

THUDM/slime

Revert "Add fallback for get_seqlen_balanced_partitions"

合并时间：2026-04-21 10:27

原文链接：http://prhub.com.cn/THUDM/slime/pull/1848

执行摘要

- 一句话：回滚序列长度平衡分区的后备机制，恢复原始分区算法。
- 推荐动作：该 PR 值得关注，因为它回滚了一个重要的内存安全机制。建议精读以理解回滚动机，并关注后续是否会有更稳健的分区方案。

功能与动机

PR 正文仅说明“Reverts THUDM/slime#1823”，未提供具体原因。结合上下文推测，可能是 PR #1823 引入的后备机制在实际运行中存在问题或不再需要，因此决定回滚到更简单的原始实现。

实现拆解

1. 移除后备分区函数：删除 _get_capped_partitions 函数，该函数原本用于在平衡分区超出 token 预算时提供 cap-aware 分区作为后备。
2. 简化 VPP 对齐逻辑：将虚拟流水线并行（VPP）的 microbatch 数量对齐逻辑从向上取整（确保对齐到每阶段组大小）改为向下取整（仅需被 vpp_size 整除），简化了计算。
3. 移除后备检查：在动态批处理路径中，移除对分区是否超出 max_tokens 的检查，不再调用后备函数，直接使用 get_seqlen_balanced_partitions 的结果。

关键文件：

- slime/backends/megatron_utils/data.py（模块 数据迭代器；类别 source；类型 core-logic；符号 _get_capped_partitions, get_data_iterator）：这是唯一变更的文件，包含了动态批处理的核心逻辑，移除后备机制直接影响内存安全性。

关键符号：_get_capped_partitions, get_data_iterator

关键源码片段

slime/backends/megatron_utils/data.py

这是唯一变更的文件，包含了动态批处理的核心逻辑，移除后备机制直接影响内存安全性。

```
def get_data_iterator(
    args: Namespace,
    model: torch.nn.Module | Sequence[torch.nn.Module],
    rollout_data: RolloutBatch,
```

```
) -> tuple[list[DataIterator], list[int]]:
    # ... 省略前部分代码 ...
    if not args.use_dynamic_batch_size:
        # 固定批处理路径
        num_microbatches = [num_local_gbs // args.micro_batch_size for _ in range(num_steps_
        per_rollout)]
        data_iterator = _generate_data_iterator(rollout_data, args.micro_batch_size)
    else:
        # 动态批处理路径
        assert args.max_tokens_per_gpu is not None
        # 计算每个 step 的 microbatch 数量
        samples = rollout_data["total_lengths"]
        num_microbatches = []
        for i in range(num_steps_per_rollout):
            start, end = i * num_local_gbs, (i + 1) * num_local_gbs
            num_microbatches.append(
                get_minimum_num_micro_batch_size(samples[start:end], args.max_tokens_per_gpu
                * cp_size)
            )
        # 全局同步最大 microbatch 数量
        num_microbatches = torch.tensor(num_microbatches, dtype=torch.int, device=torch.cuda.
        current_device())
        dist.all_reduce(num_microbatches, op=dist.ReduceOp.MAX, group=dp_group)

        if vpp_size > 1:
            # VPP 对齐逻辑：从向上取整改为向下取整，仅需被 vpp_size 整除
            num_microbatches = torch.clamp(
                num_microbatches // microbatch_group_size_per_vp_stage * microbatch_group_size_
                per_vp_stage,
                min=1,
            )

        num_microbatches = num_microbatches.tolist()

        # 平衡每个 microbatch 的序列长度
        micro_batch_indices = []
        for i, num_mbs in enumerate(num_microbatches):
            start, end = i * num_local_gbs, (i + 1) * num_local_gbs
            samples = rollout_data["total_lengths"][start:end]
            # 直接使用平衡分区，不再检查是否超出 max_tokens
            partitions = get_seqlen_balanced_partitions(samples, num_mbs, equal_size=False)
            # 调整索引偏移
            for j in range(num_mbs):
                for k in range(len(partitions[j])):
                    partitions[j][k] += start
            micro_batch_indices.extend(partitions)

        assert len(set(sum(micro_batch_indices, []))) == num_local_samples
        data_iterator = _generate_data_iterator(rollout_data, None, micro_batch_indices)
```

```
    return data_iterator, num_microbatches
```

评论区精华

该 PR 没有 review 评论，直接由作者合并。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 回归风险：移除了后备机制，如果 get_seqlen_balanced_partitions 在某些边缘情况下产生超出 GPU 内存限制的分区，可能导致 OOM 错误。
  2. 兼容性风险：VPP 对齐逻辑变更可能影响虚拟流水线并行的训练稳定性，特别是当 microbatch 数量接近边界时。
  3. 测试覆盖不足：变更未附带测试更新，无法验证回滚后逻辑的正确性。
- 影响：
  1. 对系统：简化了动态批处理逻辑，可能提升执行效率，但牺牲了内存安全性保障。
  2. 对用户：使用动态批处理的训练任务可能面临更高的 OOM 风险，需依赖 get_seqlen_balanced_partitions 的可靠性。
  3. 对团队：回滚决策表明 PR #1823 的解决方案可能不理想，团队需关注序列长度分区算法的健壮性。 - 风险标记：核心路径变更，缺少测试覆盖，内存安全风险

关联脉络

- PR #1823 Add fallback for get_seqlen_balanced_partitions: 该 PR 是本次回滚的直接对象，引入了被移除的后备机制。