PR #2100 完整报告

THUDM/slime

Remove bshd support

合并时间：2026-06-18 12:11

原文链接：http://prhub.com.cn/THUDM/slime/pull/2100

执行摘要

- 一句话：去除 bshd 数据布局支持，统一使用 THD 格式
- 推荐动作：值得精读。该 PR 展示了如何通过移除冗余布局来简化核心管线，对理解 slime 的数据布局与上下文并行实现有较大帮助。

功能与动机

PR Body 指出："This will temporarily affect vlm example and we will refactor the vlm example soon." 结合代码，之前同时支持 THD 和 BSHD 两种布局增加了维护成本和复杂性。移除 BSHD 支持后，可以简化核心数据流，减少分支判断，提高代码可读性和维护性。此外，BSHD 布局可能已经不被实际使用。

实现拆解

1. 移除命令行参数和配置校验：在 slime/utils/arguments.py 中删除 --qkv-format 参数及其取值校验，所有逻辑不再需要区分 THD 和 BSHD。
2. 重命名并清理辅助函数：在 slime_plugins/megatron_bridge/glm4v_moe.py 中，将 _thd_to_bshd 重命名为 _thd_to_batch_seq，_bshd_to_thd 重命名为 _batch_seq_to_thd，表示现在只是批量序列格式而非特殊 BSHD 布局；更新所有调用点。
3. 简化数据加载（data.py）：移除 qkv_format 参数，移除 BSHD 分支，统一使用 THD 逻辑；同样简化 cp_utils.py 中的 get_logits_and_tokens_offset_with_cp、get_sum_of_sample_mean、slice_with_cp 函数，删除 qkv_format 和 max_seq_len 参数。
4. 精简损失计算（loss.py）：移除 get_responses 中的 BSHD 处理分支和 max_seq_lens 参数，移除了 _allgather_cp_redistribute 中的相关参数。
5. 更新 actor、model 和测试文件：移除对 qkv_format 的引用和分支；修改测试中的参数；更新 VLM 示例脚本和 README。

关键文件：

- slime_plugins/megatron_bridge/glm4v_moe.py（模块 模型桥；类别 source；类型 core-logic；符号 _thd_to_bshd, _thd_to_batch_seq, _bshd_to_thd, _batch_seq_to_thd）：核心桥接层，重命名并精简了 THD↔BSHD 转换函数，移除对 BSHD 布局的显式依赖。
- slime/backends/megatron_utils/data.py（模块 数据管线；类别 source；类型 core-logic；符号 get_batch）：数据加载入口，移除 BSHD 分支和 qkv_format 参数，完全统一为 THD 逻辑。

- slime/backends/megatron_utils/loss.py（模块 损失计算；类别 source；类型 core-logic；符号 get_responses, _allgather_cp_redistribute）：损失计算与响应提取，去除 BSHD 分支和 max_seq_lens 参数，大幅精简代码。
- slime/backends/megatron_utils/cp_utils.py（模块 上下文并行；类别 source；类型 core-logic；符号 get_logits_and_tokens_offset_with_cp, get_sum_of_sample_mean, slice_with_cp）：上下文并行帮助函数，移除 qkv_format 和 max_seq_len 参数，简化偏移计算。
- slime/backends/megatron_utils/actor.py（模块 推理执行器；类别 source；类型 core-logic；符号 _get_rollout_data）：推理执行器，移除对 qkv_format 的引用和相关分支。
- slime/utils/arguments.py（模块 参数配置；类别 source；类型 configuration；符号 add_train_arguments, slime_validate_args）：参数定义与校验，删除 --qkv-format 参数及校验逻辑，移除配置入口。
- slime/backends/megatron_utils/model.py（模块 模型定义；类别 source；类型 data-contract）：模型定义，移除对 qkv_format 的引用和分支。
- tests/test_value_temperature.py（模块 值温度；类别 test；类型 test-coverage）：测试更新，适配参数移除后的变化。
- tests/test_megatron_argument_validation.py（模块 参数校验；类别 test；类型 test-coverage）：参数校验测试，移除了对 --qkv-format 的校验用例。
- examples/geo3k_vlm/run_geo3k_qwen35.sh（模块 示例脚本；类别 other；类型 core-logic）：VLM 示例启动脚本，移除 --qkv-format bshd 参数，暂时停用 BSHD。
- examples/geo3k_vlm/README.md（模块 文档；类别 docs；类型 documentation）：文档同步更新，反映参数变化。

关键符号：_thd_to_batch_seq, _batch_seq_to_thd, get_batch, get_responses, get_logits_and_tokens_offset_with_cp, get_sum_of_sample_mean, slice_with_cp, _get_rollout_data, add_train_arguments, slime_validate_args

关键源码片段

slime_plugins/megatron_bridge/glm4v_moe.py

核心桥接层，重命名并精简了 THD↔BSHD 转换函数，移除对 BSHD 布局的显式依赖。

```
# slime_plugins/megatron_bridge/glm4v_moe.py (head)

# 将 THD 格式解包为 [bs, max_seq, ...] 布局
# 之前命名为 _thd_to_bshd，现更名为 _thd_to_batch_seq，表明其通用性
def _thd_to_batch_seq(packed: torch.Tensor, cu_seqlens: torch.Tensor) -> torch.Tensor:
    """Unpack THD-format [1, T, ...] to [bs, max_seq, ...] using cu_seqlens."""
    seqlens = cu_seqlens[1:] - cu_seqlens[:-1]
    max_seq = seqlens.max().item()
    bs = len(cu_seqlens) - 1
    out = packed.new_zeros(bs, max_seq, *packed.shape[2:])
    for i, sl in enumerate(seqlens):
        out[i, :sl] = packed[0, cu_seqlens[i] : cu_seqlens[i] + sl]
```

```
    return out

# 将 [bs, max_seq, ...] 打包回 THD [1, T, ...]
def _batch_seq_to_thd(unpacked: torch.Tensor, cu_seqlens: torch.Tensor) -> torch.Tensor:
    """Pack [bs, max_seq, ...] back to THD [1, T, ...]."""
    seqlens = cu_seqlens[1:] - cu_seqlens[:-1]
    total = cu_seqlens[-1].item()
    out = unpacked.new_zeros(1, total, *unpacked.shape[2:])
    for i, sl in enumerate(seqlens):
        out[0, cu_seqlens[i] : cu_seqlens[i] + sl] = unpacked[i, :sl]
    return out

# 在 forward 中调用（原位置不变，但符号已更新）
input_ids_batch_seq = _thd_to_batch_seq(full_input_ids, cu_seqlens)
pos_batch_seq = self._compute_mrope_position_ids(input_ids_batch_seq, image_grid_thw)
pos_packed = _batch_seq_to_thd(pos_batch_seq.permute(1, 2, 0), cu_seqlens)
position_ids = pos_packed.permute(2, 0, 1).contiguous() # [3, 1, T_global]
```

slime/backends/megatron_utils/data.py

数据加载入口，移除 BSHD 分支和 qkv_format 参数，完全统一为 THD 逻辑。

```
# slime/backends/megatron_utils/data.py (head)

def get_batch(
    data_iterator: "DataIterator",
    keys: Sequence[str],
    pad_multiplier: int = 128,
    allgather_cp: bool = False,
) -> dict:
    """生成 CP 就绪的微批次，统一使用 THD 格式。

    Args:
        data_iterator: 迭代器。
        keys: 需要获取的字段列表。
        pad_multiplier: 填充倍数（默认 128）。
        allgather_cp: 是否使用 DSA 模式。
    """
    assert "tokens" in keys
    batch = data_iterator.get_next(keys)
    tokens = batch["tokens"]
    pad_token_id = 0
    pad_size = mpu.get_tensor_model_parallel_world_size() * pad_multiplier
    batch["unconcat_tokens"] = tokens

    cp_size = mpu.get_context_parallel_world_size()
    cp_rank = mpu.get_context_parallel_rank()

    # 移除 qkv_format 分支，仅保留 THD 实现
    if allgather_cp:
```

```
        cu_seqlens_list = [0]
        for t in tokens:
            cu_seqlens_list.append(cu_seqlens_list[-1] + t.size(0))
        tokens = torch.cat(tokens, dim=0)
        global_pad_size = cp_size * pad_size
        pad = (global_pad_size - tokens.size(0) % global_pad_size) % global_pad_size
        if pad != 0:
            tokens = F.pad(tokens, (0, pad), value=pad_token_id)
            cu_seqlens_list.append(cu_seqlens_list[-1] + pad)
        cu_seqlens = torch.tensor(cu_seqlens_list, dtype=torch.int, device=torch.cuda.current_
        device())
        tokens = tokens.chunk(cp_size, dim=0)[cp_rank]
    else:
        tokens = [slice_with_cp(t, pad_token_id) for t in tokens]
        cu_seqlens = [0]
        for t in tokens:
            cu_seqlens.append(cu_seqlens[-1] + t.size(0))
        tokens = torch.cat(tokens)
        pad = (pad_size - tokens.size(0) % pad_size) % pad_size
        if pad != 0:
            tokens = F.pad(tokens, (0, pad), value=pad_token_id)
            cu_seqlens.append(cu_seqlens[-1] + pad)
        cu_seqlens = torch.tensor(cu_seqlens, dtype=torch.int).cuda() * cp_size

    max_seqlen = (cu_seqlens[1:] - cu_seqlens[:-1]).max().item()
    packed_seq_params = PackedSeqParams(
        cu_seqlens_q=cu_seqlens,
        cu_seqlens_kv=cu_seqlens,
        max_seqlen_q=max_seqlen,
        max_seqlen_kv=max_seqlen,
        qkv_format="thd",
    )
    tokens = tokens.unsqueeze(0)
    batch["tokens"] = tokens
    batch["packed_seq_params"] = packed_seq_params
    # ... 后续 loss_masks 处理保持不变
    return batch
```

评论区精华

本 PR 无 review 讨论，为提交者自行决策并合并。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. VLM 示例兼容性：PR Body 明确承认会暂时影响 *example/geo3k_vlm*，后续将重构。若当前仍依赖 BSHD 格式执行 VLM 任务，将直接失败。

2. 向后兼容性：删除 --qkv-format 参数后，任何仍在配置中传递 --qkv-format bshd 的启动脚本都会报错，需移除该参数。
3. 代码简化带来的隐藏裂隙：移除大量条件分支后，THD 通路可能暴露先前被掩藏的 corner case，但测试文件已同步更新，风险可控。

- 影响：
  1. 用户：使用 --qkv-format bshd 的配置将无法工作；使用默认 THD 的用户不受影响。
  2. 系统：核心数据流分支减少，可能获得微小性能提升并降低出错概率。
  3. 团队：维护成本降低，后续开发只需关注 THD 布局；VLM 示例需要尽快重构以恢复正常使用。 - 风险标记：VLM 示例受影响，移除命令行参数需配置同步，核心路径变更

关联脉络

- PR #2081 sync from internal and cleanup: 同为清理型 PR，移除了部分冗余代码，本 PR 延续了简化后端配置的风格。
- PR #2057 Allow zero-GPU rollout router startup: 涉及相同的 arguments.py 和 actor.py 文件，共同推动后端配置简化。