PR #1969 完整报告

THUDM/slime

support --save-hf for raw mode

合并时间：2026-05-28 16:25

原文链接：http://prhub.com.cn/THUDM/slime/pull/1969

执行摘要

- 一句话：新增 raw mode 下 --save-hf 直接保存 HF 权重
- 推荐动作：建议阅读 hf_checkpoint_saver.py 中的 save_hf_model_direct 和 _SafetensorShardWriter 实现，这是典型的 Megatron → HF 权重重排和分片模式。如果团队有自定义 checkpoint 需求，可参考此设计。测试文件 test_hf_checkpoint_saver.py 也有助于理解预期行为。

功能与动机

此前只有通过 Megatron Bridge 才能将 Megatron 模型保存为 HuggingFace 格式，增加了不必要的依赖和复杂度。为了支持更轻量、灵活的保存方式，需要提供不依赖 Bridge 的直接转换路径。

实现拆解

1. 新增 hf_checkpoint_saver.py：实现 save_hf_model_direct 函数，该函数在 rank-0 上创建目标目录、清理旧权重文件、复制非权重的资产（config.json、tokenizer.json 等），然后通过 HfWeightIteratorDirect 迭代权重并写入 safetensors 分片。内部类 _SafetensorShardWriter 管理分片写入和生成模型索引文件。
2. 修改 model.py：在 save_hf_model 函数开头添加分支判断，若 args.megatron_to_hf_mode 不是 'bridge'，则调用新导入的 save_hf_model_direct 并提前返回，否则保留原有 Bridge 路径。
3. 修改三个权重迭代器（hf_weight_iterator_base.py / direct.py / bridge.py）：为 get_hf_weight_chunks 方法增加 progress_desc 参数，便于调用方自定义进度条描述。
4. 新增测试文件 test_hf_checkpoint_saver.py：使用临时目录模拟 HF 仓库结构，验证 _clear_existing_hf_weights 只删除权重而保留配置、_copy_hf_assets 复制非权重文件并跳过权重、_SafetensorShardWriter 正确写入分片并生成 index 文件。
5. 调整 CI 配置文件（pr-test.yml 及模板）和 requirements.txt，确保新代码通过 lint 和测试。

关键文件：

- slime/backends/megatron_utils/hf_checkpoint_saver.py（模块 转换保存；类别 source；类型 core-logic；符号 save_hf_model_direct, _SafetensorShardWriter, init, write）：核心实现文件，新增 save_hf_model_direct 函数和 _SafetensorShardWriter 类，实现不依

赖 Bridge 的 HF 模型保存。

- slime/backends/megatron_utils/model.py（模块 保存入口；类别 source；类型 data-contract）：保存入口文件，修改 save_hf_model 函数支持根据 megatron_to_hf_mode 选择路径，是路由新功能的核心分支。
- tests/utils/test_hf_checkpoint_saver.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_copy_hf_assets_keeps_quantized_config_and_skips_weights, test_clear_existing_hf_weights_removes_old_weight_files_only, test_safetensor_shard_writer_writes_hf_index）：新增测试文件，覆盖新实现的核心辅助函数和分片写入逻辑，保障质量。
- slime/backends/megatron_utils/update_weight/hf_weight_iterator_base.py（模块 权重迭代器；类别 source；类型 core-logic；符号 get_hf_weight_chunks）：抽象基类，修改 get_hf_weight_chunks 签名以支持 progress_desc 参数，确保接口一致。
- slime/utils/arguments.py（模块 参数配置；类别 source；类型 configuration）：新增 --megatron_to_hf_mode 参数，控制使用桥接还是直接模式，是配置入口。

关键符号：save_hf_model_direct, _SafetensorShardWriter.init, _SafetensorShardWriter.write, _SafetensorShardWriter.finalize, _clear_existing_hf_weights, _copy_hf_assets, get_hf_weight_chunks

关键源码片段

slime/backends/megatron_utils/hf_checkpoint_saver.py

核心实现文件，新增 save_hf_model_direct 函数和 _SafetensorShardWriter 类，实现不依赖 Bridge 的 HF 模型保存。

```
def save_hf_model_direct(args, rollout_id: int, model) -> None:
    '''直接保存 Megatron 模型为 HuggingFace safetensors checkpoint，绕过 Megatron
    Bridge。'''
    import torch.distributed as dist
    from transformers import AutoConfig
    from .update_weight.common import named_params_and_buffers
    from .update_weight.hf_weight_iterator_direct import HfWeightIteratorDirect

    path = Path(args.save_hf.format(rollout_id=rollout_id))
    is_save_rank = _is_global_rank_zero()
    hf_checkpoint = Path(args.hf_checkpoint).resolve()
    save_path = path.resolve()
    # 避免覆盖原始 HF checkpoint 目录
    if hf_checkpoint == save_path:
        raise ValueError('--save-hf must not point to the same directory as --hf-checkpoint')
    if not hf_checkpoint.is_dir():
        raise ValueError(f'--hf-checkpoint must be a local directory when using raw --save-hf:
        {args.hf_checkpoint}')

    setup_error = None
    if is_save_rank:
```

```
    try:
        # 在 rank 0 上创建目录、清理旧权重、复制非权重资产
        path.mkdir(parents=True, exist_ok=True)
        _clear_existing_hf_weights(path)
        _copy_hf_assets(args.hf_checkpoint, path)
    except Exception as e:
        setup_error = repr(e)
_raise_if_rank_zero_failed('prepare raw HuggingFace save directory', setup_error)

# 收集 HuggingFace 元数据（模型名称、量化配置）
metadata_error = None
payload: list[Any] = [None]
if is_save_rank:
    try:
        hf_config = AutoConfig.from_pretrained(args.hf_checkpoint, trust_remote_code=True)
        payload = [
            (
                type(hf_config).__name__.lower() if args.model_name is None else args.model_
                name,
                getattr(hf_config, 'quantization_config', None),
            )
        ]
    except Exception as e:
        metadata_error = repr(e)
_raise_if_rank_zero_failed('load HuggingFace conversion metadata', metadata_error)

if dist.is_available() and dist.is_initialized():
    dist.broadcast_object_list(payload, src=0) # 广播给所有 rank
model_name, quantization_config = payload[0]

# 初始化直接权重迭代器
hf_weight_iterator = HfWeightIteratorDirect(
    args=args,
    model=model,
    model_name=model_name,
    quantization_config=quantization_config,
)
# 收集本地的 Megatron 参数和缓冲区
megatron_local_weights = dict(named_params_and_buffers(args, model, convert_to_global_
name=True))
writer = _SafetensorShardWriter(path, enabled=is_save_rank)

# 分块转换并写入 safetensors 文件
for hf_named_tensors in hf_weight_iterator.get_hf_weight_chunks(
    megatron_local_weights, progress_desc='Save HF checkpoint'
):
    write_error = None
    try:
        writer.write(hf_named_tensors)
```

```
            except Exception as e:
                write_error = repr(e)
            _raise_if_rank_zero_failed('write raw HuggingFace weight shard', write_error)
            del hf_named_tensors
            if torch.cuda.is_available():
                torch.cuda.ipc_collect()

        finalize_error = None
        if is_save_rank:
            try:
                writer.finalize()
            except Exception as e:
                finalize_error = repr(e)
        _raise_if_rank_zero_failed('finalize raw HuggingFace checkpoint', finalize_error)

        if is_save_rank:
            logger.info('Successfully saved HuggingFace model to %s', path)
```

评论区精华

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 分布式通信风险：save_hf_model_direct 使用 broadcast_object_list 同步元数据，但仅在 dist 已初始化时执行；若未初始化可能在非 rank-0 上使用 None 作为 model_name 导致异常。不过从代码看广播后有检查。
  2. 权重转换正确性：HfWeightIteratorDirect 依赖于 named_params_and_buffers 和 convert_to_hf，这些函数在其他上下文中已有使用，但直接保存路径下可能遇到未知布局转换错误。
  3. 分片写入错误：若 writer.write 在 rank-0 上抛出异常，_raise_if_rank_zero_failed 会触发所有 rank 的异常，但 if is_save_rank 分支外的异常未传播？实际上 _raise_if_rank_zero_failed 传递错误字符串到其他 rank，应该能同步出错。但需要验证非 rank-0 能否正确识别错误。
  4. 配置兼容性：新增的 --megatron_to_hf_mode 参数未在 arguments.py 中的 patch 体现？从源码分析看 arguments.py 有修改，但未展示具体内容，可能添加了该参数。若缺失，新代码可能报错。 - 影响：对用户的影响：新增配置项 --megatron_to_hf_mode，默认行为不变（缺省可能为 'bridge'），但用户可通过设置非 bridge 值启用直接保存。现有使用 Bridge 的 workflow 不受影响。对系统的影响：引入约 200 行新代码，增加维护负担，但降低了对 Megatron Bridge 的强制依赖。长期有利于解耦。对团队的影响：需确保新路径在多种分布式配置下正确，尤其是 Pipeline/Expert parallel 场景。 - 风险标记：核心路径变更，分布式通信依赖，缺少集成测试

关联脉络

- 暂无明显关联 PR