PR #6335 完整报告

verl-project/verl

[megatron] chore: refactor to use Megatron-Bridge new APIs

合并时间：2026-05-21 10:52

原文链接：http://prhub.com.cn/verl-project/verl/pull/6335

执行摘要

- 一句话：重构 Megatron 后端以采用上游 Megatron-Bridge 新 API
- 推荐动作：建议在合并前或合并后立即验证 reviewer 指出的默认配置问题，特别是 MoE 设置和 adapter_path 访问。这些值得精读的地方在于如何平衡减少重复代码和保持行为兼容。整体设计方向正确，但需关注回归。

功能与动机

Megatron-Bridge 发布了新辅助 API（见 upstream PR #3813 和 #3866），这些 API 封装了常见的 PEFT 设置、DDP 配置和 provider 配置等模式。移植到这些 API 可以消除 verl 中的重复代码，确保与上游保持一致，并简化未来的升级。

实现拆解

1. 移除本地模型构建代码：verl/models/mcore/bridge.py 从本地定义的 LinearForLastLayer、make_value_model、freeze_moe_router 等改为直接从 megatron.bridge.training.utils.train_utils 导入，同时删除 _ensure_model_list 辅助函数。
2. 重构 PEFT 和 DDP 配置：verl/utils/megatron_utils.py 中的 make_megatron_module 函数使用 create_peft_hook 替代手动 PEFT 转换，使用 load_peft_adapter_checkpoint 替代自定义加载逻辑，并使用 create_ddp_config 替代本地 DDP 配置字典。
3. 更新 transformer 构建：verl/workers/engine/megatron/transformer_impl.py 的 _build_tf_config 方法改用 provider.configure(override_transformer_config, provider_overrides) 并传递 pre-finalize hook，替代原来直接设置 provider 属性的模式。同时调整了动态上下文并行配置的位置。
4. 清理量化补丁：完全删除 verl/utils/modelopt/megatron_qat_patch.py（394 行 monkey patch），因为上游现在原生支持 SwiGLU sharded factory 和 EP gather。更新 vllm_modelopt_patch.py 以适配上游 nvfp4_marlin_process_scales 返回元组的新格式，添加解包辅助函数。
5. 增强 checkpoint 序列化：verl/utils/checkpoint/megatron_checkpoint_manager.py 引入 _to_json_safe_config_value 等函数，递归地处理配置数据类型（torch.dtype、Enum、numpy 等）以生成安全的 JSON，同时修复了 FSDP checkpoint 加载路径使用新的 load_fsdp_dtensor_checkpoint。

6. 更新配置文件和 CI：verl/workers/config/megatron_peft.py、verl/utils/modelopt/__init__.py 等文件相应调整导入；.github/workflows/e2e_ppo_trainer_megatron_vllm.yml 微调环境变量。

关键文件：

- verl/utils/modelopt/megatron_qat_patch.py（模块 量化补丁；类别 source；类型 deletion；符号 apply_swiglu_sharded_factory_patch, _patched_apply_swiglu_sharded_factory, sh_ten_build_fn, sh_ten_merge_fn）：删除了 394 行 monkey patch，是本次重构的核心，因为这些补丁的功能已被 Megatron-Bridge 原生支持，完全移除减少了维护负担。
- verl/utils/modelopt/vllm_modelopt_patch.py（模块 量化适配；类别 source；类型 data-contract；符号 _unwrap_marlin_scale, _split_marlin_scale, _require_fp4_marlin_supported, _modelopt_dense_init_marlin）：更新了 NVFP4/Marlin 权重处理以适配上游 API 的变更，新增了 scale 解包函数，并修复了除零风险，是 QAT 工作流的关键维护。
- verl/models/mcore/bridge.py（模块 模型桥接；类别 source；类型 data-contract；符号 _ensure_model_list, LinearForLastLayer, init, forward）：从本地定义改为从 Megatron-Bridge 导入 LinearForLastLayer、make_value_model 等，删除了 150+ 行自定义实现，是模型封装层的主要简化。
- verl/utils/checkpoint/megatron_checkpoint_manager.py（模块 检查点管理；类别 source；类型 core-logic；符号 _to_json_safe_config_value, _to_json_safe_config_dict, _config_to_shallow_dict）：重构 checkpoint 加载路径，使用新的 load_fsdp_dtensor_checkpoint，并新增了递归 JSON 安全配置序列化函数，提高了配置保存的鲁棒性。
- verl/utils/megatron_utils.py（模块 训练工具；类别 source；类型 core-logic；符号 peft_pre_wrap_hook, adapter_checkpoint_hook, peft_info_hook）：使用 Megatron-Bridge 的 create_peft_hook、load_peft_adapter_checkpoint、create_ddp_config 替代本地 PEFT 和 DDP 配置逻辑，但 reviewer 指出了 adapter_path 访问方式不兼容 dict 配置的问题。
- verl/workers/engine/megatron/transformer_impl.py（模块 引擎实现；类别 source；类型 dependency-wiring）：重构 _build_tf_config，使用 provider.configure() 替代属性赋值，但 reviewer 指出缺失了 MoE 和 attention backend 的默认配置，以及动态 CP 配置仅添加在 vanilla bridge 分支。

关键符号：apply_swiglu_sharded_factory_patch, _patched_apply_swiglu_sharded_factory, sh_ten_build_fn, sh_ten_merge_fn, revert_swiglu_sharded_factory_patch, apply_ep_gather_patch, _patched_gather_from_ep_ranks, revert_ep_gather_patch, _unwrap_marlin_scale, _split_marlin_scale, _require_fp4_marlin_supported, _modelopt_dense_init_marlin, _modelopt_moe_init_marlin, _ensure_model_list, LinearForLastLayer, init, forward, make_value_model, hook, freeze_moe_router, _to_json_safe_config_value, _to_json_safe_config_dict, _config_to_shallow_dict, peft_pre_wrap_hook, adapter_checkpoint_hook, peft_info_hook

关键源码片段

verl/utils/modelopt/vllm_modelopt_patch.py

更新了 NVFP4/Marlin 权重处理以适配上游 API 的变更，新增了 scale 解包函数，并修复了除零风险，是 QAT 工作流的关键维护。

```
# 新增辅助函数：处理 nvfp4_marlin_process_scales 返回格式的变化
def _unwrap_marlin_scale(value):
    """如果 nvfp4_marlin_process_scales 返回 (processed_scale, scale_factor)，提取 processed_
    scale。"""
    return value[0] if isinstance(value, tuple) else value


def _split_marlin_scale(value):
    """如果返回值不是元组，则包装为 (value, 1.0) 以统一处理。"""
    return value if isinstance(value, tuple) else (value, 1.0)


# 在 _modelopt_dense_process_weights 中的关键修改
# 使用 _split_marlin_scale 解包上游可能返回的元组
marlin_weight_scale, scale_factor = _split_marlin_scale(
    nvfp4_marlin_process_scales(weight_scale, a_dtype=param_dtype)
)
# 全局 scale 除以 scale_factor 以保持数值一致性
marlin_weight_global_scale = (
    nvfp4_marlin_process_global_scale(weight_scale_2_max.to(torch.float32), param_dtype) /
    scale_factor
)
```

verl/models/mcore/bridge.py

从本地定义改为从 Megatron-Bridge 导入 LinearForLastLayer、make_value_model 等，删除了 150+ 行自定义实现，是模型封装层的主要简化。

```
# 头部导入现在直接从 Megatron-Bridge 的训练工具模块导入需要的组件
try:
    from megatron.bridge import AutoBridge
    from megatron.bridge.training.utils.train_utils import (
        LinearForLastLayer,
        freeze_moe_router,
        make_value_model,
    )
except ImportError:
    print("Megatron-Bridge package not found. Please install Megatron-Bridge with `pip install
    megatron-bridge`")
    raise

# 显式声明公共 API，方便使用者明确可导入符号
__all__ = [
    "AutoBridge",
```

```
    "LinearForLastLayer",
    "freeze_moe_router",
    "make_value_model",
]
```

verl/utils/checkpoint/megatron_checkpoint_manager.py

重构 checkpoint 加载路径，使用新的 load_fsdp_dtensor_checkpoint，并新增了递归 JSON 安全配置序列化函数，提高了配置保存的鲁棒性。

```
# 递归地将配置值转换为 JSON 安全的类型，处理 torch.dtype、Enum、numpy、循环引用等
_SKIP_CONFIG_VALUE = object()


def _to_json_safe_config_value(value, seen):
    # 基础类型直接返回
    if value is None or isinstance(value, str | int | float | bool):
        return value
    # torch.dtype 和 Enum 转为字符串
    if type(value) is torch.dtype or isinstance(value, Enum):
        return str(value)
    # numpy 标量转为 Python 标量
    if isinstance(value, np.generic):
        return value.item()
    # 可调用对象跳过
    if callable(value):
        return _SKIP_CONFIG_VALUE
    # 列表 / 元组递归处理，并利用 seen 检测循环引用
    if isinstance(value, list | tuple):
        value_id = id(value)
        if value_id in seen:
            return _SKIP_CONFIG_VALUE
        seen.add(value_id)
        converted = [_to_json_safe_config_value(item, seen) for item in value]
        seen.remove(value_id)
        return converted
    # 字典递归处理
    if isinstance(value, dict):
        value_id = id(value)
        if value_id in seen:
            return _SKIP_CONFIG_VALUE
        seen.add(value_id)
        converted = {}
        for key, item in value.items():
            converted_key = _to_json_safe_config_value(key, seen)
            converted_item = _to_json_safe_config_value(item, seen)
            if converted_key is not _SKIP_CONFIG_VALUE and converted_item is not _SKIP_
            CONFIG_VALUE:
                converted[str(converted_key)] = converted_item
        seen.remove(value_id)
```

```
            return converted
        return _SKIP_CONFIG_VALUE
```

评论区精华

- 在 transformer_impl.py 中，reviewer 指出 provider.configure() 不再包括关键的 MoE 设置（moe_token_dispatcher_type、moe_router_load_balancing_type）和 attention backend 默认，可能导致 MoE 模型行为变化。
- 在 megatron_utils.py 中，reviewer 提到 DDP 配置生成使用 create_ddp_config 可能丢失 Megatron-FSDP 特定的参数（如 check_for_nan_in_grad、overlap_grad_reduce）。
- 同样在 megatron_utils.py，adapter_path 通过 getattr 访问 peft_config，但 peft_config 可能是字典，导致始终为 None，无法加载 adapter checkpoint。
- 动态上下文并行配置仅添加到 vanilla bridge 分支，当使用 Megatron-Bridge provider 时动态 CP 配置被遗漏。
- 在 vllm_modelopt_patch.py 中，除法 scale_factor 和 input_global_scale 可能存在零除风险，需添加保护。
- Transformer impl 中缺乏 MoE 和注意力后端默认配置 (performance): 提交者未直接回应，但后续提交可能部分解决；最终代码未明确添加这些默认值，说明可能信赖上游默认。
- DDP 配置生成可能丢失 Megatron-FSDP 特定参数 (performance): 未明确讨论，但可能通过 overrides 参数传入。建议确认上游 create_ddp_config 是否应用了相同默认值。
- adapter_path 访问方式不兼容 dict 配置 (correctness): 尚未修复。可能导致用户无法加载 PEFT adapter。
- 动态上下文并行配置在非 vanilla bridge 分支被遗漏 (correctness): 未明确修复。要么需要同样添加到 provider_overrides，要么显式报错不支持。
- scale_factor 除零风险 (correctness): 未明确修复。代码未处理。
- 使用 __globals__访问内部符号的脆弱性 (other): 未修复。

风险与影响

- 风险：
  - 训练行为风险：缺失的 MoE 默认配置可能影响 MoE 模型的训练稳定性和性能（transformer_impl.py）。
  - 性能风险：注意力后端默认值可能不是最优，导致吞吐量下降。
  - 功能风险：PEFT adapter path 访问方式错误使得用户无法正确加载预训练 adapter（megatron_utils.py）。
  - 运行时崩溃风险：vllm_modelopt_patch.py 中的除法操作可能在零权重层导致崩溃。
  - 依赖风险：使用 __globals__ 访问内部符号，对 vLLM 版本敏感。
  - 兼容性风险：使用上游新 API 可能需要特定的 Megatron-Bridge 版本，与旧版本不兼容。
- 影响：
  - 用户：对于使用 MoE、PEFT 或动态 CP 的用户，行为可能发生变化，需要验证现有训练脚本的输出是否一致。对于使用 QAT 的用户，vllm patch 的改动可能影响量化权重的加

载和转换。

- 系统：代码量减少，维护负担降低；但对上游 API 的依赖增强。
- 团队：需要确保 Megatron-Bridge 版本约束正确，并跟进上游的更新。
- 风险标记：缺失关键默认值，PEFT 配置不兼容，除零风险，动态 CP 分支遗漏，上游 API 版本依赖

关联脉络

- PR #6318 [megatron, cfg] feat: add Qwen3.5-35B Megatron-Bridge launch script on Ascend: 修改了相同文件 verl/utils/checkpoint/megatron_checkpoint_manager.py，该 PR 重构了 checkpoint 相关功能，本 PR 进一步在此文件上使用新 API。