

PR #7061 完整报告

verl-project/verl

[ckpt] feat: add save_lora_only checkpoint support

合并时间: 2026-07-17 11:21

原文链接: <http://prhub.com.cn/verl-project/verl/pull/7061>

执行摘要

- 一句话: 新增仅保存 LoRA 适配器权重的 checkpoint 支持
- 推荐动作: 此项变更设计清晰, 向后兼容, 建议 LoRA 训练用户启用该功能。值得关注的设计决策是键模式检测 (`lora_/.adapter_`) 和向后兼容策略, 类似模式可复用于其他 adapter 场景。

功能与动机

在 LoRA 微调场景下, 完整保存模型权重 (~54 GiB) 造成巨大的存储和传输开销。该 PR 旨在只保存 LoRA 适配器权重 (~150 MiB), 大幅降低 checkpoint 体积。PR #6409 仅保存元数据 JSON, 未实现权重过滤。

实现拆解

1. 新增配置项: 在 `verl/trainer/config/config.py` 的 `CheckpointConfig` 中添加 `save_lora_only: bool = False` 字段, 并更新文档注释。
2. 基类支持: 在 `BaseCheckpointManager` (`verl/utils/checkpoint/checkpoint_manager.py`) 中添加 `should_save_lora_only` 属性 (从 config 读取) 和静态方法 `is_lora_only_state_dict` (检测 state dict 中所有键是否包含 `lora_` 或 `.adapter_` 关键字)。
3. FSDP 保存逻辑: 在 `FSDPCheckpointManager.save_checkpoint` 中, 当 `should_save_lora_only` 为 `True` 且模型包含 `peft_config` 时, 过滤 state dict 仅保留 LoRA 键; 计算并记录 LoRA 参数大小; 若过滤后无键则抛出 `ValueError`。
4. FSDP 加载逻辑: 在 `FSDPCheckpointManager.load_checkpoint` 中, 加载模型 state dict 后调用 `is_lora_only_state_dict` 判断是否为 LoRA-only checkpoint; 若是, 使用 `strict=False` 加载, 并检查 `unexpected_keys` 确保适配器配置匹配, 防止静默失败。
5. 测试与文档: 新增包含 13 个单元测试的测试文件, 覆盖默认值、配置开关、各种 state dict 模式 (`full/lora-only/mixed`) 的检测、以及 FSDP 的 `_has_lora`、加载异常路径等。同时更新了两份文档说明使用方式和效果。

关键文件:

- `tests/checkpoint_engine/test_fsdp_lora_only_checkpoint_on_cpu.py` (模块单元测试; 类别 `test`; 类型 `test-coverage`; 符号 `TestBaseCheckpointManagerLoraOnly`, `_patch_dist`, `test_should_save_lora_only_default`, `test_should_save_lora_only_false`): 新增 13 个单元测试, 覆盖 `save_lora_only` 的配置、state dict 检测、保存过滤和加载异常

路径。

- verl/utils/checkpoint/fsdp_checkpoint_manager.py (模块 检查点实现; 类别 source; 类型 core-logic; 符号 _has_lora, load_checkpoint, save_checkpoint) : 核心实现文件: 在保存时过滤 LoRA 权重, 在加载时自动检测 LoRA-only checkpoint 并合并; 新增 _has_lora 方法。
- verl/utils/checkpoint/checkpoint_manager.py (模块 检查点基类; 类别 source; 类型 core-logic; 符号 should_save_lora_only, is_lora_only_state_dict) : 基类新增 should_save_lora_only 属性和 is_lora_only_state_dict 静态方法, 为所有 checkpoint 管理器提供统一接口。
- verl/trainer/config/config.py (模块 训练配置; 类别 source; 类型 configuration; 符号 save_lora_only) : 在 CheckpointConfig 中添加 save_lora_only 布尔字段, 控制是否仅保存 LoRA 权重。
- docs/advance/checkpoint.rst (模块 开发者文档; 类别 docs; 类型 documentation) : 新增 LoRA-only Checkpoints 章节, 说明使用方式和效果。
- docs/examples/config.rst (模块 示例文档; 类别 docs; 类型 documentation) : 添加 save_lora_only 配置选项的示例说明。

关键符号: BaseCheckpointManager.should_save_lora_only,
BaseCheckpointManager.is_lora_only_state_dict, FSDPCheckpointManager._has_lora,
FSDPCheckpointManager.load_checkpoint, FSDPCheckpointManager.save_checkpoint

关键源码片段

tests/checkpoint_engine/test_fsdp_lora_only_checkpoint_on_cpu.py

新增 13 个单元测试, 覆盖 save_lora_only 的配置、state dict 检测、保存过滤和加载异常路径。

```
import pytest
from verl.utils.checkpoint.checkpoint_manager import BaseCheckpointManager
```

```
class TestBaseCheckpointManagerLoraOnly:
    """Tests for LoRA-only checkpoint properties on BaseCheckpointManager."""

    @pytest.fixture(autouse=True)
    def _patch_dist(self, monkeypatch):
        # 模拟分布式环境, 避免需要真实初始化
        import torch.distributed
        monkeypatch.setattr(torch.distributed, "get_rank", lambda: 0)
        monkeypatch.setattr(torch.distributed, "get_world_size", lambda: 1)
        monkeypatch.setattr(torch.distributed, "barrier", lambda: None)

    def test_should_save_lora_only_true(self):
        # 当 checkpoint_config 中的 save_lora_only 为 True 时, 属性应返回 True
        mgr = self._make_manager(checkpoint_config={"save_lora_only": True})
        assert mgr.should_save_lora_only is True
```

```

def test_is_lora_only_state_dict_lora(self):
    # 当所有键包含 "lora_" 时，应判定为 LoRA-only 状态
    state_dict = {
        "base_model.model.model.layers.0.self_attn.q_proj.lora_A.weight": None,
        "base_model.model.model.layers.0.self_attn.q_proj.lora_B.weight": None,
    }
    assert BaseCheckpointManager.is_lora_only_state_dict(state_dict) is True

    @staticmethod
    def _make_manager(checkpoint_config):
        # 辅助方法，创建带有 mock 的 BaseCheckpointManager 实例
        class MockModel:
            pass
        class MockOptimizer:
            pass
        return BaseCheckpointManager(
            model=MockModel(), optimizer=MockOptimizer(),
            lr_scheduler=None, processing_class=None,
            checkpoint_config=checkpoint_config,
        )

```

verl/utils/checkpoint/fsdp_checkpoint_manager.py

核心实现文件：在保存时过滤 LoRA 权重，在加载时自动检测 LoRA-only checkpoint 并合并；新增 `_has_lora` 方法。

```

def save_checkpoint(self, local_path: str, hdfs_path: str = None, global_step: int = -1):
    # ... 省略其他部分
    if self.should_save_model:
        model_state_dict = self.model.state_dict()
        # 当开启 save_lora_only 且模型包含 peft_config 时，过滤 state dict
        if self.should_save_lora_only and self._has_lora():
            n_total = len(model_state_dict)
            # 只保留键中包含 "lora_" 或 ".adapter_" 的参数 (LoRA 权重)
            model_state_dict = {
                k: v for k, v in model_state_dict.items()
                if "lora_" in k or ".adapter_" in k
            }
            # 若过滤后 state dict 为空，则抛出异常，避免保存空文件
            if not model_state_dict:
                raise ValueError(
                    f"save_lora_only is True and the model has a peft_config, "
                    f"but no LoRA/adapter parameters were found in the state dict. "
                    f"Total params checked: {n_total}."
                )
            # 计算参数总字节数 (兼容 ShardedTensor)
            lora_bytes = 0
            for v in model_state_dict.values():
                if hasattr(v, "numel") and hasattr(v, "element_size"):
                    lora_bytes += v.numel() * v.element_size()

```

```

elif hasattr(v, "local_shards"):
    # 兼容 PyTorch < 2.1 中的 ShardedTensor
    for shard in v.local_shards():
        lora_bytes += shard.tensor.numel() * shard.tensor.element_size()
lora_mib = lora_bytes / 1024**2
log_with_rank(
    f"LoRA-only save: {len(model_state_dict)}/{n_total} params ({lora_mib:.1f} MiB)",
    rank=self.rank, logger=logger, log_only_rank_0=True,
)
torch.save(model_state_dict, model_path)
# ... 后续日志

```

verl/utils/checkpoint/checkpoint_manager.py

基类新增 `should_save_lora_only` 属性和 `is_lora_only_state_dict` 静态方法，为所有 checkpoint 管理器提供统一接口。

```

@property
def should_save_lora_only(self) -> bool:
    # 如果 checkpoint_config 为 None，则返回 False（安全默认值）
    if not self.checkpoint_config:
        return False
    # 从 config 对象中获取 save_lora_only 字段，默认 False
    return getattr(self.checkpoint_config, "save_lora_only", False)

@staticmethod
def is_lora_only_state_dict(state_dict: dict) -> bool:
    # 空 dict 或 None 视为普通 checkpoint
    if not state_dict:
        return False
    # 当且仅当所有键都包含 "lora_" 或 ".adapter_" 时才认为是 LoRA-only
    return all("lora_" in k or ".adapter_" in k for k in state_dict)

```

评论区精华

- 加载时缺少 unexpected keys 验证: gemini-code-assist[bot] 指出使用 `strict=False` 加载 LoRA-only checkpoint 时，若目标模型没有 LoRA 适配器，`load_state_dict` 会静默忽略所有键，导致加载失败无提示。作者随后在 `load_checkpoint` 中增加了对 `result.unexpected_keys` 的检查，若有意外键则抛出 `ValueError`。
- ShardedTensor 属性缺失及空 state dict 风险: gemini-code-assist[bot] 指出在 `SHARDED_STATE_DICT` 模式下，state dict 值可能是 `ShardedTensor`，没有 `numel/element_size` 属性，计算大小时会 `AttributeError`；同时过滤后 state dict 可能为空，保存空文件造成问题。作者修复了大小计算逻辑（兼容 `ShardedTensor` 的 `local_shards`），并增加了空 state dict 时抛出 `ValueError` 的保护。
- 加载 LoRA-only checkpoint 时缺少对 unexpected keys 的验证 (correctness): 作者在后续提交中向 `load_checkpoint` 增加了对 `unexpected_keys` 的检查，若存在则抛出 `ValueError`。

- 计算 ShardedTensor 大小时可能 AttributeError 以及空 state dict 保护 (correctness): 作者修复了大小计算逻辑, 使用 hasattr 检查并处理 ShardedTensor 的 local_shards; 同时增加了空 state dict 时抛出 ValueError 的保护。

风险与影响

- 风险:
 - 键命名依赖: LoRA 键检测依赖 lora_ 和 .adapter_ 关键字, 若用户使用自定义前缀可能导致检测失败, 回退到完整保存 (安全)。
 - 后端局限: 目前仅 FSDPCheckpointManager 实现了 _has_lora 和过滤逻辑, 其他后端 (如 Megatron) 不生效。若用户在其他后端开启 save_lora_only, should_save_lora_only 返回 True 但保存逻辑无改动, 可能仍保存全量权重, 造成误导。
 - 空 state dict 异常: 当 LoRA 键全部被过滤掉时会抛出 ValueError, 防止保存无效 checkpoint, 但可能中断训练, 需用户确认 LoRA 配置。
- 影响:
 - 用户: LoRA 训练用户可显著减少 checkpoint 存储和网络带宽占用, 加载时自动合并权重, 对训练流程透明。
 - 系统: 新增配置项不影响已有配置, 旧 checkpoint 不受影响, 完全向后兼容。
 - 团队: 需维护此功能并扩展至其他 checkpoint 后端; 新增 13 个单元测试覆盖主要路径, 降低回归风险。
 - 风险标记: LoRA 键命名依赖, 仅 FSDP 后端生效, 空 state dict 异常打断训练

关联脉络

- PR #6409 [ckpt] feat: save lora train meta: 前序工作, 仅保存 LoRA 训练的元数据 (rank, alpha 等) 到 JSON, 未实现权重过滤。本 PR 是其实质性扩展, 实现了 LoRA 权重的实际保存与加载。