PR #6512 完整报告

verl-project/verl

[fsdp, model] feat: per-unit LoRA summon, FSDP1/2 compatibility, and strip-modules support

合并时间：2026-06-18 11:37

原文链接：http://prhub.com.cn/verl-project/verl/pull/6512

执行摘要

- 一句话：重构 FSDP+LoRA 分层召唤，降低显存并修复死锁
- 推荐动作：该 PR 值得所有深入使用 FSDP+LoRA 训练的用户精读，尤其是 layered_summon_lora_params 的设计思路和 wrap 策略防护逻辑。建议结合具体模型进行回归测试，验证显存与训练稳定性。对于重复 all-gather 的风险，可在升级后仍保持监控，必要时在内部 fork 中进一步优化。

功能与动机

为了支持大规模多阶段模型（如 Qwen3-Omni-30B-A3B Thinker）的 RL 微调，需要降低 FSDP 下 LoRA 参数收集的显存峰值，并解决 NCCL 死锁和 FSDP state_dict hooks 崩溃等问题。PR body 明确指出：'All changes are model-agnostic and do not introduce any dependency on vllm-omni'。

实现拆解

步骤 1：重写 layered_summon_lora_params

- 文件：verl/utils/fsdp_utils.py
- 关键变更：移除旧的 __prefix_submodules 和硬编码前缀；改为遍历整个模块树的 named_modules()，对每个 FSDP 单元（通过 fsdp_version 检测）依次调用 summon_full_params 或 DTensor.full_tensor()。峰值显存从整个模型降为最大 FSDP 单元。
- FSDP1/FSDP2 兼容：通过 fsdp_version() 判断是否需使用 summon_full_params 上下文；对 FSDP2 的 DTensor 参数直接调用 full_tensor()。
- 跳过退化单元：过滤掉不包含 lora_* 可训练参数的 FSDP 单元，避免 state_dict() 崩溃。

步骤 2：调整 LoRA wrap 策略

- 文件：verl/utils/fsdp_utils.py
- 问题：同时使用 lambda 策略和基于大小的策略会创建嵌套 FSDP 单元，在变长输入 use_remove_padding 下 all-gather 顺序不一致，导致 NCCL 死锁。
- 修复：当 min_num_params > 0 时禁用 lambda 策略，避免混合策略冲突。

步骤 3：支持模块剥离与 dtype 转换

- 文件：verl/workers/engine/fsdp/transformer_impl.py
- _verl_strip_modules：在模型加载后，删除 module._verl_strip_modules 中列出的子模块（如 talker、code2wav），使其不占用 GPU 内存。
- LoRA dtype 转换：在 _build_lora_module 中，将 fp32 的 LoRA 适配器参数转为与基模型一致的 bf16，避免 FSDP flatten group 内的 dtype 不匹配错误。

步骤 4：修复 monkey_patch 中 text_config 的获取

- 文件：verl/models/transformers/monkey_patch.py
- 问题：多阶段模型（如 Qwen3-Omni）的 text_config 不直接暴露在 model.config.text_config，而是在嵌套结构中。
- 修复：使用 get_text_config() 替代直接属性访问，并保留向后兼容 fallback。

关键文件：

- verl/utils/fsdp_utils.py（模块 FSDP 工具；类别 source；类型 core-logic；符号 layered_summon_lora_params, collect_lora_params, get_wrap_policy）：核心改动：重写 layered_summon_lora_params、FSDP1/2 兼容、wrap 策略修复、collect_lora_params 调整
- verl/workers/engine/fsdp/transformer_impl.py（模块 FSDP 引擎；类别 source；类型 core-logic；符号 _build_module, _build_lora_module）：支持子模块剥离 (_verl_strip_modules) 和 LoRA dtype 自动转换
- verl/models/transformers/monkey_patch.py（模块 模型补丁；类别 source；类型 data-contract；符号 apply_monkey_patch）：修复多阶段模型 text_config 属性访问的兼容性

关键符号：layered_summon_lora_params, collect_lora_params, get_wrap_policy, _build_module, _build_lora_module, apply_monkey_patch

关键源码片段

verl/utils/fsdp_utils.py

核心改动：重写 layered_summon_lora_params、FSDP1/2 兼容、wrap 策略修复、collect_lora_params 调整

```
def layered_summon_lora_params(fsdp_module) -> OrderedDict:
    """
    逐个 FSDP 单元收集 LoRA 参数，避免一次收集全部模型导致显存爆炸。
    遍历所有 named_modules()，对每个 FSDP unit（依据 fsdp_version）单独召唤。
    使用 clean_prefix 去除内部 wrapping 前缀，保持与 downstream loader 兼容。
    注意：如果父单元和子单元都是 FSDP 单元，父单元的 named_parameters 会
    递归包含子单元的参数，导致重复 all-gather；作者声称已修复此问题，但
    当前实现仍存在理论上的重复收集风险（参见 review 讨论）。
    """
    lora_params = OrderedDict()
    peft_model = getattr(fsdp_module, "_fsdp_wrapped_module", fsdp_module)

    for name, submodule in fsdp_module.named_modules():
```

```
        # 跳过 root，因为 root summon 会收集整个模型，违背分层目的
        if name == "":
            continue
        # 只处理 FSDP 包裹的 unit（FSDP1 或 FSDP2）
        if fsdp_version(submodule) == 0:
            continue

        # 去掉内部的 _fsdp_wrapped_module. 前缀，使参数名与下游期望一致
        clean_prefix = name.replace("_fsdp_wrapped_module.", "")
        # 跳过容器型单元（.model / .layers），它们没有直接参数
        if clean_prefix.endswith(".model") or clean_prefix.endswith(".layers"):
            continue

        if fsdp_version(submodule) == 1:
            # FSDP1 必须在 summon 上下文中才能获取完整参数
            with FSDP.summon_full_params(submodule):
                params = dict(submodule.named_parameters(recurse=True))
        else: # FSDP2: 通过 DTensor.full_tensor() 获取，无需 summon
            params = {}
            for n, p in submodule.named_parameters(recurse=True):
                if isinstance(p, DTensor):
                    params[n] = p.full_tensor()
                elif isinstance(p, torch.Tensor):
                    params[n] = p # 已经 gathered

        # 只保留 LoRA 参数
        lora_suffix = "lora_A.default.weight"
        for n, p in params.items():
            clean_name = clean_prefix + "." + n
            clean_name = _strip_fsdp_wrapper(clean_name)
            if clean_name.endswith(lora_suffix) or "lora_" in n:
                lora_params[clean_name] = cpu_if_needed(p)

    return lora_params
```

verl/workers/engine/fsdp/transformer_impl.py

支持子模块剥离 (_verl_strip_modules) 和 LoRA dtype 自动转换

```
# _build_module 中的剥离逻辑（位于 module 加载之后）
if self.model_config.model_type == "language_model":
    module = auto_class.from_pretrained(...)

    # 从模型属性 _verl_strip_modules 读取待删除的子模块列表
    _strip_list = getattr(module, "_verl_strip_modules", [])
    for attr in _strip_list:
        if hasattr(module, attr):
            delattr(module, attr)
            logger.info(f"Stripped unused sub-module '{attr}' to reduce memory")

# _build_lora_module 中的 dtype 转换逻辑
```

```
module = get_peft_model(module, LoraConfig(**lora_config))

# FSDP 要求同一个 flat group 内所有参数 dtype 一致;
# 如果 LoRA 适配器默认是 fp32，需转换到基模型 dtype（通常是 bf16）
base_dtype = next((p.dtype for p in module.parameters() if not p.requires_grad), None)
if base_dtype is not None:
    mismatched = [p for p in module.parameters() if p.requires_grad and p.dtype != base_dtype]
    if mismatched:
        logger.info(f"Casting {len(mismatched)} LoRA adapter params from "
                    f"{mismatched[0].dtype} to {base_dtype}")
        for param in mismatched:
            param.data = param.data.to(base_dtype)
```

评论区精华

- 嵌套 FSDP 单元重复收集问题：gemini-code-assist[bot] 指出 layered_summon_lora_params 在遍历父 FSDP 单元时会递归收集子单元参数，导致重复 all-gather 和 OOM 风险。作者回复已修复，但最终代码仍保留 submodule.named_parameters(recurse=True) 模式，重复问题是否完全解决需要进一步验证。
- state_dict hooks 冲突：该 bot 建议在 collect_lora_params 中显式构造 state_dict 传参，避免 get_peft_model_state_dict 默认调用 state_dict() 导致 hooks 冲突。SamitHuang 认为 'not critical, but cheap and safer'，但最终代码未明确修改该部分。
- 插件加载机制：早期提交添加了自定义插件加载器，但 wuxibin89 指出 verl 已存在 VERL_USE_EXTERNAL_MODULES hook，重复实现导致后续移除，最终 PR 未包含插件相关改动。
- import 位置与注释风格：SamitHuang 要求将 import 移至文件头部并简化注释，作者最终遵循了建议。
  - 嵌套 FSDP 单元重复收集导致 OOM (performance): 作者回复 'fixed.'，但最终关键代码未显式过滤已收集的子单元参数，重复问题可能未完全解决。SamitHuang 表示同意并催促作者处理。
  - collect_lora_params 未显式传递 state_dict 可能触发 hooks 冲突 (correctness): SamitHuang 认为 'not critical, but cheap and safer'，但最终代码未包含该修改。作者未回复是否采纳。
  - 自定义插件加载器与已有 VERL_USE_EXTERNAL_MODULES 重复 (design): 作者在后续提交中移除了自定义插件加载器，回归使用现有机制。
  - 代码风格：import 位置和注释简洁性 (style): 作者最终遵循了这些风格建议。

风险与影响

- 风险:
  - 重复 all-gather 风险：当前 layered_summon_lora_params 实现未显式跳过已处理的子单元，仍可能触发重复 all-gather，在大型 MoE 模型上实际存在 OOM 风险（引用 reviewer 指出的 OOM 问题）。

  - wrap 策略行为变更：当 min_num_params > 0 时禁用 lambda 策略，现有用户若依赖 lambda+size 混合策略可能发现 wrap 粒度改变，需验证训练收敛性。
  - strip_modules 依赖模型属性：_verl_strip_modules 是模型自定义属性，如果模型未设置或设置错误，不会生效或可能删除错误模块。
  - dtype 转换覆盖不全：仅对 requires_grad=True 的 params 进行 dtype 转换，若基模型部分参数不需梯度且 dtype 不一致，可能导致 FC group dtype 不匹配。
  - FSDP 版本检测：fsdp_version(submodule) 函数可能无法覆盖所有 PyTorch 版本，需在 PyTorch 2.4~2.6+ 上充分测试。
- 影响：
  - 用户视角：所有使用 FSDP+LoRA 训练的用户将受益于显存降低和训练稳定性提升。特别是训练 30B+ MoE 或包含多阶段子模块的模型时，之前可能遇到 OOM 或死锁的任务现在可正常运行。
  - 系统视角：峰值显存下降，可支持更大模型或更大 batch size 在同一 GPU 集群上训练。
  - 团队视角：改动了 FSDP 核心工具函数，后续 FSDP 相关开发需要兼容此次引入的新模式；同时清理了若干废弃标记，降低了技术债务。
  - 影响范围：直接修改 3 个文件，但影响所有 FSDP 后端训练流程（包括 PPO、GRPO 等）。影响程度中等偏大。
  - 风险标记：嵌套 FSDP 重复收集，wrap 策略行为变更，strip_modules 依赖模型属性，dtype 转换覆盖不全，FSDP 版本检测依赖

关联脉络

- 暂无明显关联 PR