PR #7283 完整报告

verl-project/verl

[fsdp,veomni] fix: backfill missing state for DSD optimizer checkpoint

合并时间：2026-08-06 18:05

原文链接：http://prhub.com.cn/verl-project/verl/pull/7283

执行摘要

- 一句话：回填扁平化 DSD 检查点缺失优化器状态，修复恢复崩溃
- 推荐动作：建议精读。改动小但处于训练恢复核心路径，设计上巧妙复用 optimizer.state_dict() 的初值来对齐 allow_partial_load 语义。值得关注的是缺少单元测试覆盖，可参考 tests/utils/ckpt/test_lora_custom_object_save_on_cpu.py 的写法，补一个 CPU 上的扁平化 state dict 回填校验。

功能与动机

PR body 说明：扁平化 DSD 检查点只持久化已拥有优化器状态的参数（即收到过梯度的参数），从未收到梯度的参数——如 DeepSeek-V4 稀疏注意力 indexer，其 forward 只返回 top-k 索引——在 torch unflatten 时被变成空 dict 而不是跳过，最终在 Adam.__setstate__ 中失败。回填新初始化优化器的状态正是模拟 DCP allow_partial_load 的行为：这些参数以默认状态恢复。

实现拆解

1. 背景与入口：该修复落在 verl/utils/checkpoint/fsdp_checkpoint_manager.py 的 FSDPCheckpointManager.load_checkpoint 优化器加载分支。此前该分支直接 torch.load 后调用 self.optimizer.load_state_dict(optimizer_state_dict)，遇到扁平化 DSD 格式会崩溃。
2. 新增回填方法：新增 _backfill_optimizer_state。首先通过顶层键 state / param_groups 区分普通优化器 state dict 与扁平化 DSD 格式：普通格式直接透传，因为 torch 本身容忍无状态参数；扁平化格式则取当前新初始化优化器的 state_dict() 作为基准，计算检查点中缺失的键。
3. 合并与日志：当存在缺失键时，返回 {**current_state_dict,**state_dict}，检查点中已有的状态优先，缺失条目保留初始值，并通过 log_with_rank 在 rank 0 打印缺失数量和示例键，便于排查。
4. 调用点接线：在 load_checkpoint 的优化器分支中，torch.load 之后、load_state_dict 之前插入 optimizer_state_dict = self._backfill_optimizer_state(optimizer_state_dict)。
5. 配套改动：无测试文件、无配置变更，纯增量 35 行。建议后续在 tests/utils/ckpt/ 下补充 CPU 上的扁平化 state dict 回填测试。

关键文件：

- verl/utils/checkpoint/fsdp_checkpoint_manager.py（模块 检查点；类别 source；类型 core-logic；符号 _backfill_optimizer_state）：唯一变更文件，新增 _backfill_optimizer_state 并在 load_checkpoint 优化器加载分支调用，修复扁平化 DSD 检查点恢复崩溃。

关键符号：_backfill_optimizer_state

关键源码片段

verl/utils/checkpoint/fsdp_checkpoint_manager.py

唯一变更文件，新增 _backfill_optimizer_state 并在 load_checkpoint 优化器加载分支调用，修复扁平化 DSD 检查点恢复崩溃。

```
    def _backfill_optimizer_state(self, state_dict: dict) -> dict:
        """回填扁平化 DSD 优化器检查点中缺失的状态条目。

        VeOmni 的 ``MultiOptimizer`` 走 ``torch.distributed.checkpoint.state_dict``
        且 ``flatten_optimizer_state_dict=True`` 保存时，只持久化收到过梯度的参数；
        从未收到梯度的参数（如 DeepSeek-V4 稀疏注意力 indexer）在 unflatten 后会被
        torch 变成空 dict，触发 ``Adam.__setstate__`` 报错。这里用新初始化优化器的
        状态补齐缺失条目，等价于 DCP ``allow_partial_load``：缺失参数从默认状态恢复。
        该方法由 ``load_checkpoint`` 在优化器加载分支中调用。
        """
        # 普通优化器 state dict（含 "state" / "param_groups" 顶层键）：
        # torch 本身就容忍无状态参数，直接透传，无需回填。
        if "state" in state_dict or "param_groups" in state_dict:
            return state_dict

        # 扁平化格式：以当前（刚初始化）优化器的 state_dict 为基准，
        # 它的键集合代表所有可训练参数。
        current_state_dict = self.optimizer.state_dict()
        # 防御：若当前优化器返回普通格式，说明两侧结构不一致，保持原样。
        if "state" in current_state_dict or "param_groups" in current_state_dict:
            return state_dict

        # 检查点中缺失的键：这些参数从未收到梯度，需要回填初始状态。
        missing_keys = [key for key in current_state_dict if key not in state_dict]
        if not missing_keys:
            return state_dict

        log_with_rank(
            f"{len(missing_keys)} optimizer state entries are absent from the checkpoint "
            f"and keep their initial value, e.g. {missing_keys[:5]}",
            rank=self.rank,
            logger=logger,
            log_only_rank_0=True,
        )
        # 合并：检查点已有的状态优先（后覆盖），缺失条目自动退回初始值。
        return {**current_state_dict, **state_dict}
```

```
if self.should_load_optimizer:
    remote_optim_path = os.path.join(local_path, f"optim_world_size_{self.world_size}_rank_
    {self.rank}.pt")
    local_optim_path = copy_to_local(remote_optim_path)
    optimizer_state_dict = torch.load(local_optim_path, weights_only=False)
    # 关键：在 load_state_dict 之前回填扁平化 DSD 检查点缺失的优化器状态，
    # 避免 unflatten 空 dict 导致 Adam.__setstate__ 崩溃。
    optimizer_state_dict = self._backfill_optimizer_state(optimizer_state_dict)
    self.optimizer.load_state_dict(optimizer_state_dict)
```

评论区精华

该 PR 无 review 评论与讨论线程。设计权衡体现在方法 docstring 中：用刚初始化的优化器补齐缺失条目，刻意对齐 DCP allow_partial_load 的语义，避免在恢复阶段为这些参数临时构造默认状态；同时用 state / param_groups 键区分格式，确保普通优化器检查点路径零行为变化。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 缺少测试覆盖：改动位于分布式恢复核心路径，但无对应测试文件；若 DSD 扁平化键空间与当前优化器 state_dict() 不一致，{**current_state_dict,**state_dict} 可能引入多余或冲突的键。
  2. 静默恢复风险：从未收到梯度的参数会被静默用初始状态恢复；若某个参数本应有状态却因存储问题缺失，该逻辑会掩盖真实数据问题。
  3. 兼容性：防御分支仅检查顶层键 state / param_groups，若后续 torch DCP 版本改变扁平化格式的键名或结构，missing_keys 计算可能失效。 - 影响：影响范围集中在 FSDP 与 VeOmni 引擎的优化器检查点恢复路径，受益最大的场景是 DeepSeek-V4 这类含 sparse-attention indexer（前向只返回 top-k 索引、从不产生梯度）的模型训练恢复；对普通（非扁平化）优化器检查点无行为变化，因为回填前已短路。团队侧降低了 VeOmni 扁平化保存方案的恢复门槛，也为未来其他零梯度参数模型（如稀疏专家、固定 embedding）提供了默认恢复语义。 - 风险标记：缺少测试覆盖，合并依赖两侧键空间一致，静默回填可能掩盖数据缺失

关联脉络

- PR #7117 [ckpt] fix: save base model's code, not the PeftModel wrapper's, in FSDP checkpoints: 同样修改 verl/utils/checkpoint/fsdp_checkpoint_manager.py，属于 FSDP 检查点保存 / 恢复正确性修复线。
- PR #7272 [fsdp,veomni] feat: support pad_to_length to reduce jit compile time: 同属 FSDP/VeOmni 引擎路径，显示该模块近期在性能与稳定性上的持续投入。
- PR #7243 [veomni] fix: preserve GPT-OSS weights without expert parallelism: VeOmni 权重 / 检查点导出修复，与本 PR 同属 VeOmni 检查点数据面。