PR #7109 完整报告

verl-project/verl

[recipe] feat: Retain model output from tinker forward backward

合并时间：2026-07-24 09:56

原文链接：http://prhub.com.cn/verl-project/verl/pull/7109

执行摘要

- 一句话：Tinker 前向反向保留模型输出供服务器使用
- 推荐动作：建议精读 FSDP 引擎中通过 non-tensor 数据传递标志的设计模式，以及 MoE 参数转换中按模型类型分发的策略。测试覆盖充分，值得参考。

功能与动机

Tinker Worker 需要获取前向反向的输出（如 log_probs）以便返回给调用方。然而，FSDPEngine.forward_backward_batch 在训练后总是丢弃 model_output 以防止 OOM。本 PR 允许 Tinker 明确选择保留输出，而不影响标准训练路径。同时修复了 GPT-OSS 模型在权重同步时因 MoE 参数展开导致的崩溃问题。

实现拆解

1. FSDP 引擎添加 return_model_output 支持：在 verl/workers/engine/fsdp/transformer_impl.py 的 forward_backward_batch 方法中，从 data 读取 non-tensor 字段 return_model_output（默认 False）。当该字段为 False 时，保持原有行为，丢弃 model_output；当为 True 时，保留 model_output 并在结果字典中包含它。
2. Tinker Worker 显式启用保留：在 verl/workers/engine_workers_tinker.py 的 forward_backward 方法中，在调用 engine 之前向 data 设置 return_model_output=True，并移除原来在 mp 源 rank 上删除 model_output 的代码。
3. vLLM MoE 参数展开适配 GPT-OSS：在 verl/workers/rollout/vllm_rollout/vllm_rollout.py 中，_iter_vllm_compatible_moe_params 新增 model_type 参数。如果 model_type == "gpt_oss"，则跳过 3D 专家张量的展开，直接 yield 原始 packed 权重，避免 vLLM 的 GPT-OSS 加载器在 TP 分片时因维度不匹配出错。
4. TeacherModel 懒加载配置：在 verl/experimental/teacher_loop/teacher_model.py 中，_initialize_llm_servers 方法不再直接实例化 HFModelConfig，而是使用 OmegaConf.create 创建一个 DictConfig，包含 _target_ 引用。每个 rollout server 在自己的进程中解析该配置，从而将 HDFS 模型的拷贝分散到各节点，避免在中央节点产生临时文件。
5. 添加测试覆盖：新增两个 CPU 测试文件：tests/workers/test_engine_return_model_output_on_cpu.py 验证 Tinker 请求 model_output 和 FSDP 尊重标记；tests/workers/rollout/test_vllm_moe_param_converter_on_cpu.py 验证 Qwen 和

GPT-OSS 的 MoE 参数转换正确。修改 tests/models/test_engine.py 以确保兼容。

关键文件：

- tests/workers/test_engine_return_model_output_on_cpu.py（模块 测试；类别 test；类型 test-coverage；符号 test_tinker_forward_backward_requests_model_output, forward_backward_batch, test_fsdp_forward_backward_honors_return_model_output）：新测试文件，验证 TinkerWorker 和 FSDP 引擎正确保留 model_output。
- tests/workers/rollout/test_vllm_moe_param_converter_on_cpu.py（模块 测试；类别 test；类型 test-coverage；符号 _collect, test_qwen_moe_packed_weights_are_expanded_per_expert, test_gpt_oss_packed_weights_are_not_expanded）：新测试文件，验证 Qwen 和 GPT-OSS 的 MoE 参数转换逻辑。
- verl/workers/rollout/vllm_rollout/vllm_rollout.py（模块 vLLM 适配；类别 source；类型 core-logic；符号 _iter_vllm_compatible_moe_params）：核心修改：_iter_vllm_compatible_moe_params 增加 model_type 参数，处理 GPT-OSS 的 MoE 兼容性。
- verl/workers/engine/fsdp/transformer_impl.py（模块 FSDP 引擎；类别 source；类型 core-logic；符号 forward_backward_batch）：FSDP 引擎关键修改：根据 return_model_output 标记决定是否保留 model_output。
- verl/workers/engine_workers_tinker.py（模块 Tinker 工作器；类别 source；类型 core-logic；符号 forward_backward）：Tinker Worker 显式设置 return_model_output 标记并移除旧的 model_output 删除。
- verl/experimental/teacher_loop/teacher_model.py（模块 教师模型；类别 source；类型 data-contract；符号 _initialize_llm_servers）：TeacherModel 配置懒加载，避免中央节点解析 HDFS 路径。
- tests/models/test_engine.py（模块 测试；类别 test；类型 test-coverage）：测试适配：确保现有测试与新数据流兼容。

关键符号：FSDPEngine.forward_backward_batch, TinkerTrainingWorker.forward_backward, _iter_vllm_compatible_moe_params, ServerAdapter.update_weights, TeacherModelManager._initialize_llm_servers

关键源码片段

tests/workers/test_engine_return_model_output_on_cpu.py

新测试文件，验证 TinkerWorker 和 FSDP 引擎正确保留 model_output。

```
# 测试 TinkerForwardBackward 是否向 data 添加 return_model_output=True
def test_tinker_forward_backward_requests_model_output():
    captured = {}

    def forward_backward_batch(data, **kwargs):
        captured["return_model_output"] = tu.get_non_tensor_data(data, key="return_model_
        output", default=None)
        return {}
```

```
    engine = SimpleNamespace(
        train_mode=lambda **kwargs: nullcontext(),
        forward_backward_batch=forward_backward_batch,
        is_mp_src_rank_with_outputs=lambda: False,
    )
    worker = SimpleNamespace(
        loss_fn=lambda: None,
        engine=engine,
        engine_config=SimpleNamespace(
            forward_only=False,
            use_dynamic_bsz=False,
            max_token_len_per_gpu=128,
            micro_batch_size_per_gpu=1,
            use_fused_kernels=False,
        ),
        model_config={},
    )

    result = TinkerTrainingWorker.forward_backward(worker, TensorDict({}, batch_size=[]))

    assert result is None
    assert captured["return_model_output"] is True


# 参数化测试验证 FSDP 引擎在 return_model_output=True 时保留 output，False 时丢弃
@pytest.mark.parametrize(("return_model_output", "expected"), [(False, False), (True, True)])
def test_fsdp_forward_backward_honors_return_model_output(monkeypatch, return_model_
output, expected):
    data = TensorDict({"loss_mask": torch.ones(1)}, batch_size=[1])
    if return_model_output:
        tu.assign_non_tensor(data, return_model_output=True)

    loss = torch.tensor(1.0, requires_grad=True)
    model_output = {"log_probs": torch.tensor([-1.25])}
    engine = SimpleNamespace(
        ulysses_sequence_parallel_size=1,
        scaler=None,
        get_data_parallel_group=lambda: None,
        get_data_parallel_size=lambda: 1,
        forward_step=lambda micro_batch, loss_function, forward_only: (
            loss,
            {"model_output": model_output.copy(), "loss": 1.0, "metrics": {}},
        ),
    )

    monkeypatch.setattr(torch.distributed, "all_reduce", lambda *args, **kwargs: None)
    monkeypatch.setattr("verl.workers.engine.fsdp.transformer_impl.get_device_id", lambda:
    "cpu")
    monkeypatch.setattr(
```

```
        "verl.workers.engine.fsdp.transformer_impl.prepare_micro_batches",
        lambda data, **kwargs: ([data], None),
    )
    monkeypatch.setattr(
        "verl.workers.engine.fsdp.transformer_impl.postprocess_batch_func",
        lambda output_lst, **kwargs: output_lst[0],
    )

    result = FSDPEngine.forward_backward_batch(engine, data, loss_function=lambda: None,
    forward_only=False)

    assert ("model_output" in result) is expected
    if expected:
        assert torch.equal(result["model_output"]["log_probs"], model_output["log_probs"])
```

tests/workers/rollout/test_vllm_moe_param_converter_on_cpu.py

新测试文件，验证 Qwen 和 GPT-OSS 的 MoE 参数转换逻辑。

```
import asyncio
import torch

from verl.workers.rollout.vllm_rollout.vllm_rollout import _iter_vllm_compatible_moe_params


def _collect(weights, model_type):
    async def collect():
        return [item async for item in _iter_vllm_compatible_moe_params(weights, model_type)]
    return asyncio.run(collect())


# 验证 Qwen MoE: packed 3D 参数被展开为每个专家的独立 2D 权重
def test_qwen_moe_packed_weights_are_expanded_per_expert():
    gate_up = torch.randn(2, 6, 8) # shape: [num_experts, 2*intermediate, hidden]
    down = torch.randn(2, 8, 3) # shape: [num_experts, hidden, intermediate]

    converted = _collect(
        [
            ("model.layers.0.mlp.experts.gate_up_proj", gate_up),
            ("model.layers.0.mlp.experts.down_proj", down),
        ],
        "qwen3_moe",
    )

    # 预期输出包含每个专家的独立 key，shape 为 (intermediate, hidden) 等
    assert [name for name, _ in converted] == [
        "model.layers.0.mlp.experts.0.gate_proj.weight",
        "model.layers.0.mlp.experts.0.up_proj.weight",
        "model.layers.0.mlp.experts.1.gate_proj.weight",
        "model.layers.0.mlp.experts.1.up_proj.weight",
```

```
        "model.layers.0.mlp.experts.0.down_proj.weight",
        "model.layers.0.mlp.experts.1.down_proj.weight",
    ]
    assert [tensor.shape for _, tensor in converted] == [
        (3, 8),
        (3, 8),
        (3, 8),
        (3, 8),
        (8, 3),
        (8, 3),
    ]


# 验证 GPT-OSS: packed 3D 参数保持原样，不展开
def test_gpt_oss_packed_weights_are_not_expanded():
    gate_up = torch.randn(2, 8, 6)
    down = torch.randn(2, 3, 8)
    weights = [
        ("model.layers.0.mlp.experts.gate_up_proj", gate_up),
        ("model.layers.0.mlp.experts.down_proj", down),
    ]

    converted = _collect(weights, "gpt_oss")

    assert [name for name, _ in converted] == [name for name, _ in weights]
    assert converted[0][1] is gate_up
    assert converted[1][1] is down
    assert converted[0][1].shape == (2, 8, 6)
    assert converted[1][1].shape == (2, 3, 8)
```

评论区精华

在审查中，Luosuu 对 teacher_model.py 中使用 OmegaConf 延迟加载配置的方式提出疑问，标注了 @wuxibin89 询问 'how should we handle this properly?'。目前该讨论未展开，PR 随后被批准。

- TeacherModel 配置懒加载的处理方式 (question): PR 被 wuxibin89 批准，讨论未展开但被视为可接受。

风险与影响

- 风险：
  1. 内存风险：启用 return_model_output 后，FSDP 引擎会保留完整的 model_output（如 log_probs），在微批次累积时可能导致 OOM。但该功能仅由 Tinker Worker 显式启用，不影响标准训练。
  2. GPT-OSS 兼容性：跳过 MoE 展开依赖 model_type 匹配，如果未来 vLLM 版本改变 GPT-OSS 加载器的行为，可能导致权重同步错误。

3. TeacherModel 延迟配置：使用 OmegaConf 的 _target_ 可能绕过一些配置校验，如果配置结构变化可能静默失败。 - 影响：直接影响 Tinker 实验性训练的用户，使其能够获取每个 token 的 log_probs 用于蒸馏或后处理。修复了 GPT-OSS + MoE 模型在 vLLM 权重同步时的崩溃问题，影响相关模型用户。TeacherModel 的改动无功能变化，属于内部优化。总体向后兼容，默认行为不变。 - 风险标记：内存压力风险，GPT-OSS 兼容性，配置校验绕过

关联脉络

- 暂无明显关联 PR