PR #2134 完整报告

THUDM/slime

fix: handle empty colocated weight buckets

合并时间：2026-06-29 15:45

原文链接：http://prhub.com.cn/THUDM/slime/pull/2134

执行摘要

- 一句话：修复空 colocated weight buckets 导致 crash 的问题
- 推荐动作：该 PR 修复了一个重要的分布式同步 Bug，设计简洁且测试覆盖良好。建议合并后团队关注相关路径（update_weight_from_tensor.py、update_weight_from_distributed.py）是否存在类似假设，预防回归。同时，新增的单元测试值得作为后续类似功能的参考。

功能与动机

在 PP/EP/MoE 模型布局下，TP rank 可能没有某个 chunk 的 HF tensors，原代码仍构建空 bucket 导致同步前 crash。需要让空贡献被视为有效。关联 Issue: EazyReal/slime#3。

实现拆解

1. 修改 _send_to_colocated_engine 中 supports_multi_dtypes 分支的条件，当 hf_named_tensors 为空时 converted_named_tensors_by_dtypes 设为空字典而非 {"dtype": []}。
2. 修改 gather 后的 source rank 处理：用 max(len(tensors) for tensors in serialized_named_tensors) 替代之前假设所有 rank 有相同 dtype 数的 len(serialized_named_tensors[0])，遍历所有 bucket 索引，对于缺失的 rank 填充由 _empty_flattened_tensor_data() 生成并序列化的空 tensor 数据。
3. 新增辅助函数 _empty_flattened_tensor_data()，返回 {"flattened_tensor": torch.empty(0, dtype=torch.uint8, device=torch.cuda.current_device()), "metadata": []}。
4. 新增测试文件 tests/test_empty_colocated_weight_bucket.py，通过 fake 类模拟 FlattenedTensorBucket、序列化器和远程调用，覆盖空 bucket 与非空混合情况。
5. 更新 CI 配置文件 pr-test.yml 及模板 pr-test.yml.j2，将新测试加入无 GPU 测试矩阵。

关键文件：

- slime/backends/megatron_utils/update_weight/update_weight_from_tensor.py（模块 Megatron 工具；类别 source；类型 core-logic；符号 _empty_flattened_tensor_data, _send_to_colocated_engine）：核心源码改动：修改了 _send_to_colocated_engine 函数，新增 _empty_flattened_tensor_data 辅助函数，修复空 bucket 问题。

- tests/test_empty_colocated_weight_bucket.py（模块 权重同步测试；类别 test；类型 test-coverage；符号 _FakeFlattenedTensorBucket, _FakeMultiprocessingSerializer, _FakeRemoteMethod, _FakeEngine）：新增 204 行测试，通过 fake 类模拟分布式环境，覆盖空 bucket、混合 bucket 等场景，确保修复正确性。
- .github/workflows/pr-test.yml（模块 CI 配置；类别 infra；类型 infrastructure）：CI 配置文件，将新测试加入矩阵，确保自动运行。
- .github/workflows/pr-test.yml.j2（模块 CI 配置；类别 infra；类型 infrastructure）：CI 模板文件，对应修改。

关键符号：_send_to_colocated_engine, _empty_flattened_tensor_data

关键源码片段

slime/backends/megatron_utils/update_weight/update_weight_from_tensor.py

核心源码改动：修改了 _send_to_colocated_engine 函数，新增 _empty_flattened_tensor_data 辅助函数，修复空 bucket 问题。

```
# 当 hf_named_tensors 为空时，converted_named_tensors_by_dtypes 应变为空字典
if getattr(FlattenedTensorBucket, "supports_multi_dtypes", False):
    converted_named_tensors_by_dtypes = {"dtype": hf_named_tensors} if hf_named_tensors else
    {}
else:
    converted_named_tensors_by_dtypes = {}
    for name, tensor in hf_named_tensors:
        dtype = tensor.dtype
        if dtype not in converted_named_tensors_by_dtypes:
            converted_named_tensors_by_dtypes[dtype] = []
        converted_named_tensors_by_dtypes[dtype].append((name, tensor))

# 后续 gather 后，source rank 用 max 计算 bucket 数，缺失时填充
if dist.get_rank() == ipc_gather_src:
    num_buckets = max(len(tensors) for tensors in serialized_named_tensors)
    empty_serialized_tensor = None
    for i in range(num_buckets):
        serialized_tensors_for_dtype = []
        for tensors in serialized_named_tensors:
            if i < len(tensors):
                serialized_tensors_for_dtype.append(tensors[i])
                continue
            # 对于缺少 bucket 的 rank，填充空序列化 tensor
            if empty_serialized_tensor is None:
                empty_tensor_data = _empty_flattened_tensor_data()
                long_live_tensors.append(empty_tensor_data)
                empty_serialized_tensor = MultiprocessingSerializer.serialize(empty_tensor_data,
                output_str=True)
            serialized_tensors_for_dtype.append(empty_serialized_tensor)
        kwargs = {
```

```
            "serialized_named_tensors": serialized_tensors_for_dtype,
            "load_format": "flattened_bucket",
            "weight_version": str(weight_version),
        }
        refs.append(ipc_engine.update_weights_from_tensor.remote(**kwargs))

# 新增辅助函数
def _empty_flattened_tensor_data():
    return {
        "flattened_tensor": torch.empty(0, dtype=torch.uint8, device=torch.cuda.current_device()),
        "metadata": [],
    }
```

tests/test_empty_colocated_weight_bucket.py

新增 204 行测试，通过 fake 类模拟分布式环境，覆盖空 bucket、混合 bucket 等场景，确保修复正确性。

```
# 测试文件中的关键 fake 类定义
class _FakeFlattenedTensorBucket:
    supports_multi_dtypes = True

    def __init__(self, *, named_tensors=None, flattened_tensor=None, metadata=None):
        # 模拟真实 FlattenedTensorBucket 的行为：空 named_tensors 导致错误
        if named_tensors is not None:
            if not named_tensors:
                raise ValueError("Cannot create empty tensor bucket")
            self._flattened_tensor = ("flattened", tuple(name for name, _ in named_tensors))
            self._metadata = tuple(name for name, _ in named_tensors)
            return
        self._flattened_tensor = flattened_tensor
        self._metadata = metadata

    def get_flattened_tensor(self):
        return self._flattened_tensor

    def get_metadata(self):
        return self._metadata

# 后续测试函数通过 monkeypatch 替换依赖，验证三种场景：
# - 全空 bucket：所有 rank 的 hf_named_tensors 为空，应无 ref 返回。
# - 混合 bucket：部分 rank 有 tensor，部分为空，source rank 应正确填充并发送一致数量的 bucket。
# - 非空 bucket：作为回归测试，确保原有逻辑不变。
```

评论区精华

审阅者 zhuzilin 评论 "nice catch!"，表示认可修复。无其他技术讨论。

- 审阅者认可修复 (other)：无异议，直接合并。

风险与影响

- 风险：
  1. 分布式假设变更：原来假设所有 rank 有相同数量的 dtype bucket，现在允许不同数量。依赖旧假设的其他代码路径可能受影响。
  2. 空 tensor 创建：_empty_flattened_tensor_data 在 cuda 设备上创建大小为 0 的 tensor，通常无害，但可能触发某些环境下的资源分配。
  3. 测试覆盖：通过 monkeypatch 模拟依赖，可能遗漏真实环境中的边缘情况。 - 影响：直接影响是修复了分布式权重同步在特定模型配置（PP/EP/MoE）下的 crash，提高了系统鲁棒性。对简单 TP 配置无影响。影响范围中等，仅涉及 colocated engine 更新路径。开发团队现在可以支持更灵活的模型切分，而无需担心空 bucket 导致失败。 - 风险标记：分布式 collective 假设变更，新创建 CUDa 空 tensor, 测试依赖 mock

关联脉络

- PR #2143 Fix parallel update_from_disk in megatron server: 同为 megatron_utils 模块的 bugfix，涉及分布式权重更新路径。
- PR #2102 Support top_p mask: 修改了 megatron_utils 相关文件，扩展了权重更新逻辑。