PR #7324 完整报告

verl-project/verl

[ckpt, fsdp] feat: sharded delta weight sync for the TorchTitan engine

合并时间：2026-08-21 10:15

原文链接：http://prhub.com.cn/verl-project/verl/pull/7324

执行摘要

- 一句话：TorchTitan 引擎接入分片 delta 权重同步，最高提速 17.2×
- 推荐动作：值得精读。三个设计决策尤其值得学习：(1) 用数学分析而非特判接纳 _StridedShard——右到左切割只置换 block 归属而非交错 block，compute_local_shape_and_global_offset 已返回置换后偏移；(2) 用 slot table 而非 probe-driven 转换器解决 lockstep gather 下的命名一致性问题；(3) gather 组的「变化 Shard 维、固定其他维」构造避免 HSDP 副本浪费，并按 rank 集合缓存保证 dist.new_group 的建组顺序一致。若要在 verl 中为新的 trainer 引擎接入 delta 同步，本 PR 是很好的实现模板。

功能与动机

delta_sharded checkpoint 引擎只同步发生变化的权重元素，要求训练引擎按 rank 导出本地分片与放置元数据；TorchTitan 引擎此前只有全量导出，选择 delta_sharded 会直接抛 NotImplementedError。PR body 用基准数据说明收益：Qwen3-8B 在 32 卡 FSDP2 下 delta_sharded 为 3.34 s 而 nccl 全广播为 34.26 s（10.3×），Qwen3-30B-A3B MoE 在 FSDP2×EP8 下为 8.07 s vs 43.43 s（5.4×）。此外，不修复 EFSDP>1 丢专家的问题，delta 的 seed（全量导出）本身不可信；不修复 HSDP 下的 mesh bug，任何 HSDP 运行都会在 worker 初始化阶段崩溃。

实现拆解

1. 新增分片导出入口（transformer_impl.py）

- TorchTitanEngine.get_per_tensor_param_shard() 遍历 self.module 的 state_dict()，通过内层 _shard() 完成「搬到 device → 转 bf16 → to_local() → 展平」，产出 (hf_name, local_flat_bf16, ShardSpec) 三元组；ShardSpec.from_param() 记录 mesh 与 placements，接收端据此推导全局 flat 坐标。
- get_per_tensor_param_delta_shard() 借助 verl.workers.engine.utils.hf_delta_export 把本次分片与 _delta_shard_snap 快照做逐字节差分，产出最终 HF 坐标的稀疏 delta 条目；快照每次导出时刷新，任何 rank 都不持有全量模型快照。
- 默认实现上移到 BaseEngine（只需分片导出加 per-parameter entry builder），删除 FSDPEngine 中两处完全相同的拷贝；Megatron 保留自己的 override（差分对象不是固定快照），veomni 保留 _hf_delta_entry，TorchTitanEngine 自带实现以处理专家堆栈的

slot 拆分。

2. HF 命名对齐与专家堆栈

- _to_hf_named_params() 把原 get_per_tensor_param 里的 sd_adapter.to_hf() 与 lm_head.weight 补回逻辑抽成共享方法，保证全量与分片导出按名字精确配对——名字只出现在一边会静默把权重冻结在 seed 值。
- _expert_stack_slots() 利用 sd_adapter.from_hf_map 结构性识别 fused expert stack（一个 torchtitan key 对应带两个 {} 占位的 HF 模板），整体导出堆栈并在 spec.hf_slots 中枚举每个专家的 HF 张量，避免 to_hf() 只产出本地专家导致的 lockstep 命名不一致。
- _hf_entry_row_slots() 处理 dim-0 identity 拆分：行 *e* 就是 full[*e*]，用一次 searchsorted 从全局索引恢复每个专家的计数，规避 probe-driven 转换器的复杂度。

3. 放置推导扩展（spec.py）

- 新增 _shard_dim()：torch 2.13 中 _StridedShard 迁到 C++ 且不再继承 Shard，p.is_shard() 会漏判；该函数同时对两种写法给出答案。
- 新增 _assert_even_strided()：拒绝 strided 偏移公式无法表达的 uneven cut。
- 新增 _shard_dims_group() 取代 _flattened_mesh_group()：变化 Shard 维、固定其他维构造 gather 组，按 rank 集合缓存（dist.new_group 是集合通信，所有 rank 以相同顺序建组），避免 HSDP 下 replicate 度被计入 wire 字节与 rank 0 的 staging 缓冲。

4. 边界防护与两个预存 bug 修复

- _assert_shard_export_supported() 在导出边界以明确信息拒绝 PP：各 stage 持有不相交切片，导出顺序在 rank 间不一致，无法满足 lockstep gather。
- _get_data_parallel_mesh() 改用 parallel_dims.get_optional_mesh("loss") 获取 1-D 扁平 loss mesh，修复 HSDP 下 get_data_parallel_rank()/get_data_parallel_group() 遇到 2-D mesh 崩溃的问题。
- get_per_tensor_param()（全量导出）改为先对整个 DTensor 专家堆栈做 full_tensor() 再拆分，修复 EFSDP>1 时先 to_hf() 拆分再在 *ep* 组内 all-gather 导致静默丢失 50% 专家的 bug。

5. 测试、兼容与文档配套

- tests/special_distributed/test_sharded_delta_gather.py 扩展 2D/3D mesh 用例：FSDP×TP（_StridedShard）、EFSDP×EP、HSDP×TP/EP、replicate 度 4 的 sharp case，并新增断言「gather 组大小等于 Shard 维大小的乘积」。
- sglang_rollout.py 对 LocalSerializedTensor 的 import 增加 SGLang 0.5.16+ 的兼容分支。
- docs/advance/delta_weight_sync.md 与 docs/workers/torchtitan_workers.rst 更新支持矩阵、设计说明与实测性能表。

关键文件：

- verl/workers/engine/torchtitan/transformer_impl.py（模块 训练引擎；类别 source；类型 core-logic；符号 _hf_entry_row_slots, _to_hf_named_params, _expert_stack_slots, _hf_delta_entry）：核心实现文件：新增分片导出与 delta 差分入口，处理 HF 命名对齐与

专家堆栈 slot table，同时修复 EFSDP 丢专家与 HSDP mesh 两个预存 bug

- verl/workers/engine/spec.py（模块 放置推导；类别 source；类型 core-logic；符号 _shard_dim, _assert_even_strided, _shard_dims_group, _flattened_mesh_group）：共享放置推导逻辑：_shard_dim / _assert_even_strided / _shard_dims_group 使 _StridedShard 与 HSDP 多 Shard 维成为一等公民，是所有 delta 后端的共同基础设施
- verl/workers/rollout/sglang_rollout/sglang_rollout.py（模块 回放端；类别 source；类型 compat-adjust；符号 _update_weights_delta_flush）：delta 应用路径的 SGLang 0.5.16+ 兼容 import，保证 TorchTitan 与 SGLang 共环境运行时不受 LocalSerializedTensor 迁移影响
- tests/special_distributed/test_sharded_delta_gather.py（模块 测试覆盖；类别 test；类型 test-coverage；符号 _run_case, main）：直接覆盖 placement 数学的分布式测试，扩展 TP/EP/HSDP 组合用例与 gather 组大小断言，是引擎级测试删除后的主要防线
- docs/advance/delta_weight_sync.md（模块 文档；类别 docs；类型 documentation）：支持矩阵、设计说明与实测性能数据的权威文档，明确 TorchTitan 的布局支持、PP 拒绝原因与 HSDP×EP 的上游依赖
- docs/workers/torchtitan_workers.rst（模块 文档；类别 docs；类型 documentation）：TorchTitan 引擎能力列表同步，列出 delta_sharded 支持的并行组合

关键符号：get_per_tensor_param_shard, get_per_tensor_param_delta_shard, _to_hf_named_params, _expert_stack_slots, _hf_entry_row_slots, _assert_shard_export_supported, _shard_dim, _assert_even_strided, _shard_dims_group, derive_dtensor_placement

关键源码片段

verl/workers/engine/torchtitan/transformer_impl.py

核心实现文件：新增分片导出与 delta 差分入口，处理 HF 命名对齐与专家堆栈 slot table，同时修复 EFSDP 丢专家与 HSDP mesh 两个预存 bug

```
def get_per_tensor_param_shard(self, **kwargs):
    """Yield this rank's local shard ``(hf_name, local_flat_bf16, ShardSpec)`` instead of the full tensor."""
    self._assert_shard_export_supported()
    raw = {}
    for module in self.module:
        raw.update(module.state_dict())

    # 专家堆栈必须整体导出并携带 slot table：to_hf() 只会保留本地拥有的专家，
    # 而 lockstep gather 要求所有 rank 对“当前参数是哪个”给出同一答案
    stacks = {}
    for name, param in raw.items():
        slots = self._expert_stack_slots(name, param)
        if slots is not None:
            stacks[name] = slots
    params = self._to_hf_named_params({k: v for k, v in raw.items() if k not in stacks})
    device = get_device_id() # 开启 offload 策略时本地分片位于 CPU
```

```
    from ..spec import ShardSpec

    def _shard(param):
        # 先把参数搬到 device 再转 bf16：混合精度下 fp32 主权重保留在优化器，
        # 线路上只传输 bf16 分片，差分时才做类型对齐
        p = param.to(device, non_blocking=True)
        if p.is_floating_point():
            p = p.to(torch.bfloat16, non_blocking=True)
        local = p.to_local() if isinstance(p, DTensor) else p
        return local.reshape(-1)

    def _gen():
        # 稠密参数：按 HF 名字逐个导出，ShardSpec 携带 mesh 与 placements，
        # 供接收端 derive_dtensor_placement 推导 flat 坐标
        for name, param in params.items():
            yield name, _shard(param), ShardSpec.from_param(param)
        # 专家堆栈：以 torchtitan 原始名字为 key，spec.hf_slots 记录
        # 每个专家的 HF 张量名与形状，接收端按 slot 拆分为逐专家 delta
        for name, slots in stacks.items():
            spec = ShardSpec.from_param(raw[name])
            spec.hf_slots = slots
            yield name, _shard(raw[name]), spec

    return _gen(), None
```

verl/workers/engine/spec.py

共享放置推导逻辑：_shard_dim / _assert_even_strided / _shard_dims_group 使 _StridedShard 与 HSDP 多 Shard 维成为一等公民，是所有 delta 后端的共同基础设施

```
def _shard_dim(p) -> Optional[int]:
    """返回该 placement 切割的张量维，不切割则返回 None。

    torch 2.13 中 _StridedShard 已迁到 C++ 且不再继承 Shard，p.is_shard() 会漏判；
    _StridedShard 与 Shard 一样是单 block 切割，只是右到左的切割顺序置换 rank 与 block
    的归属，而不是交错 block 内部。
    """
    if p.is_shard():
        return int(p.dim)
    return int(p.dim) if type(p).__name__ == "_StridedShard" else None


def derive_dtensor_placement(spec: ShardSpec) -> tuple[int | BlockPlacement, bool,
Optional[ProcessGroup]]:
    """推导本 rank 的 (place, contributes, gather_group) 三元组。

    place 供 translate_flat_indices 做分片内 flat 坐标到全量坐标的映射；
    contributes 标记副本 rank（Replicate 维坐标非 0）以空 delta 参与 lockstep；
    gather_group 决定向哪些 rank 收集分片。
    """
```

```
    import torch.distributed as dist

    assert spec.place is None # 显式 place 的 spec 由调用方分发，不会走到这里

    if spec.mesh is None:
        # 未分片：本地张量就是完整参数，无需 gather
        return 0, (dist.get_rank() == 0 if dist.is_initialized() else True), None

    placements = spec.placements
    # 用 _shard_dim 而非 p.is_shard()，否则 torch 2.13 下 _StridedShard 会被静默漏进
    # "无 shard 维"分支，mesh 维从 gather 组中消失
    shard_dims = [d for d, p in enumerate(placements) if _shard_dim(p) is not None]
    _assert_even_strided(spec, placements)

    coord = spec.mesh.get_coordinate()
    contributes = True
    if coord is not None:
        # 任一 Replicate 维坐标非 0 的 rank 不贡献数据，只以空 delta 保持 lockstep
        for d, p in enumerate(placements):
            if p.is_replicate() and coord[d] != 0:
                contributes = False
                break

    if not shard_dims:
        return 0, contributes, None

    local_shape, global_offset = compute_local_shape_and_global_offset(spec.full_shape, spec.
    mesh, list(placements))
    place = BlockPlacement(tuple(local_shape), tuple(global_offset), tuple(spec.full_shape))

    if len(shard_dims) == 1:
        # 单 Shard 维：该维子组天然只含贡献 rank，副本被跳过
        return place, contributes, spec.mesh.get_group(mesh_dim=shard_dims[0])
    # 多 Shard 维：_shard_dims_group 变化 Shard 维、固定其他维，
    # 避免 HSDP 下 replicate 度被计入 wire 字节与 rank 0 的 staging 缓冲
    return place, contributes, _shard_dims_group(spec.mesh, shard_dims)
```

评论区精华

外部 review 只有 wuxibin89 的 APPROVED，两条 review 评论均为作者 attack204 的自评说明。最有价值的反馈藏在 commit message 里：ec0f35b「drop the torchtitan delta export test」明确写道 "Review feedback. It needed 8 GPUs and a torchtitan install, so no CI job could have picked it up"——8 GPU 加 torchtitan 依赖使引擎级导出测试无法被任何 CI 任务拾取，最终删除该测试，保留直接覆盖 placement 数学的 test_sharded_delta_gather。作者在 spec.py 第 211 行的自评指出 "FSDP2 + TP is a _StridedShard, so we should modify the limitation"，即旧代码把 _StridedShard 当作「非单 block」拒绝，而实际上它只是置换 rank 归属的 block，由此驱动了 _shard_dim() 的引入与文档支持矩阵的同步更新。

- FSDP2+TP 的 _StridedShard 支持与 limitation 更新 (design): 引入 _shard_dim() 接纳 _StridedShard，并同步更新 delta_weight_sync.md 与 torchtitan_workers.rst 的支持矩阵
- 8 GPU 引擎级导出测试的可运行性 (testing): 删除 torchtitan delta 导出测试，保留直接覆盖 placement 数学的 test_sharded_delta_gather；引擎级覆盖依赖后续 CI 基建
- _hf_entry_row_slots 的专家堆栈命名转换语义 (question): 纯解释性评论，无代码变更；转换语义由 dim-0 identity 的 searchsorted 拆分解法确定

风险与影响

- 风险:
  1. torch 版本敏感：_shard_dim() 与 _assert_even_strided() 的语义依赖 torch 2.13 中 _StridedShard 的行为；同时 torchtitan 引擎本身需要 torch≥2.12（DataParallelMeshDims），而 SGLang ≤0.5.17 锁定 torch 2.11，仅在 SGLang main（torch 2.13）上收敛，环境组合较脆弱。
  2. 引擎级 e2e 测试缺失：删除 8 GPU 导出测试后，torchtitan 引擎的分片导出路径没有 CI 覆盖，只有 placement 数学的单元测试；PR body 也注明 HSDP+EP“unit-tested; no end-to-end row”。
  3. 依赖上游修复：HSDP×EP 需要 torchtitan 侧 MoE state dict adapter 修复（to_hf() 在读 placement.dim 前未检查类型），修复前该组合在 dcp_load 阶段即崩溃。
  4. 共享逻辑影响面：spec.py 的 derive_dtensor_placement() 是 FSDP/veomni/Megatron 共享的放置推导，本次改动为向后兼容的扩展（不再拒绝 _StridedShard），但任何回归都会波及其他 delta 后端。 - 影响：对 torchtitan 引擎用户，delta_sharded 从不可用到可用，跨节点 32 卡场景权重同步从 34 s 级降到 3 s 级，MoE 模型从 43 s 级降到 8 s 级；HSDP 用户额外获得 _get_data_parallel_mesh() 修复（此前任何 HSDP 运行都会在 worker 初始化崩溃）。对 nccl 全量同步用户，EFSDP>1 时全量导出静默丢失专家的预存 bug 被修复，属于正确性提升。对团队而言，17 个 commit 完整展示了从「拒绝 TP/EP」到逐步放开的设计演进、负控制验证与大量实测，文档沉淀了支持矩阵与基准数据，为后续其他引擎接入 delta 同步提供了可复用的模板。 - 风险标记：依赖 torch 2.13 行为，引擎级 e2e 测试缺失，HSDP×EP 依赖上游修复，共享放置推导逻辑改动

关联脉络

- PR #7291 [ckpt] feat: Node-local multi-sender broadcast in NCCL checkpoint engine: 同一权重同步优化方向的前序工作：NCCL 引擎多发送者中继使全量广播提速约 3 倍；本 PR 将 delta 稀疏同步推广到 torchtitan 引擎，两者共享 checkpoint 引擎与权重同步架构
- PR #7407 [megatron,veomni] feat: use torch.int16 for routed_experts: 同样深度改动 spec.py 的 ShardSpec / derive_dtensor_placement 基础设施（veomni 的 _hf_delta_entry、to_hf_chunk 探针机制），本 PR 的 slot table 方案正是对该探针机制的补充与简化
- PR #7422 [rollout] fix: preserve dummy load_format in disaggregated rollout: 分离式 rollout 的权重接收路径修复，与 delta_sharded 的 SGLang receive/apply 链路（

sglang_rollout.py 的 _update_weights_delta_flush）直接相关