PR #7181 完整报告

verl-project/verl

[megatron] feat: delta_sharded on Megatron-Bridge param mappings (TP+EP, hybrid-Mamba)

合并时间：2026-07-31 10:39

原文链接：http://prhub.com.cn/verl-project/verl/pull/7181

执行摘要

- 一句话：Megatron 后端接入分片 delta 权重同步
- 推荐动作：值得精读。核心看点：(1) comm-stubbed probe 设计——把进程组的 size/rank 语义与通信语义分离，用 NaN 哨兵穿过真实 megatron_to_hf 代码得到 HF 坐标，替代手写索引数学；(2) 幂等性验证扫描 _verify_dense 对多阶段 loader（如 Mamba 的 A = -exp(A_log)）按最终状态而非逐次 copy_ 比对的判断；(3) 测试分层——TP>1 差分 oracle + 真 sglang 闭环 + CPU 单测构成对 probe 假设的完整回归防线。建议关注 PP/LoRA 的显式报错边界，以及将差分测试纳入 CI 的后续工作。

功能与动机

让 Megatron 后端加入分片 delta 权重同步契约。追踪 issue #7060 的路线图中，FSDP + EP 的 shard 导出已由 #7144 与 #7085 完成，mcore 是下一块拼图。PR body 开宗明义：'The mcore backend joins the sharded-delta contract (#7144)'，且实现 'rides Megatron-Bridge's own per-param machinery — no legacy converters, no hand-written index math'。动机有两层：一是让 Megatron 用户获得 8–10× 的权重同步加速（235B 规模下稳态从 NCCL 全量广播的 235.0–237.6 s/step 降到 23.6 s/step，seed 同步成本低于一次 NCCL 步骤）；二是不再为每个新模型手写转换器与索引数学——probe 机制让 Mamba 的 TP 解交错、GQA 融合等带真实数值语义的转换保持位级精确。

实现拆解

1. 导出索引构建（一次性、懒加载）：verl/workers/engine/megatron/transformer_impl.py 新增 _mcore_export_index()，首次调用时经 delta_export.build_export_index() 基于 AutoBridge.get_conversion_tasks 枚举全部参数映射，为每个参数生成几何 spec 与 form-探针，并缓存至 delta_export_index / delta_export_by_name / delta_slot_cache。构建处对 vanilla_bridge 与 peft_cls 断言——旧式桥与 LoRA 直接拒绝。
2. comm-stubbed probe（核心机制）：新增 delta_export.py（307 行）。_ProbeGroup 保留真实 size()/rank()，任何真实通信通过 __getattr__ 抛错；make_probe 把 bridge 的四个通信助手（gather_from_tp_ranks / gather_from_ep_ranks[_scale] / PP 广播的 pp_size == 1 快路径）stub 为本地合成。本 rank 局部 shard 以 NaN 缓冲喂入真实 megatron_to_hf，非 NaN 幸存者即该 rank 在最终 HF 坐标的贡献。_warm_lazy_mappings 强制 AutoMapping 懒 delegate 在探针外完成实例化（否则首次探针内使用会捕获真实进程组）。_nan_block 按 (shape, dtype, device) 池化 NaN 占位块，

消解 235B 下 Opt-A 回归（+25–40%，恢复至 23.6 s）。

3. 引擎接口与内存策略：MegatronEngine 新增 get_per_tensor_param_shard()（spec 只携带线上合并组，几何全部由探针负责）、_hf_delta_entry() 与 get_per_tensor_param_delta_shard()；类属性 delta_pin_snapshots = False——整组 pin 快照在 30B/235B 下耗尽 cudaHostAlloc 池并表现为无关分配的 CUDA OOM。base.py 与 engine/utils.py 将 pin 决策收敛为引擎类级属性、prime_delta_snapshots 改为显式 pin 参数（review 驱动，见 commit 0b3e9745）。
4. 发送端校验与内存回收：delta_checkpoint_engine.py 新增 verify_every 引擎参数——每 K 次稳态同步在同一 receive 会话内追加完整 seed 重放（_publish_values_flush 携带 verify / is_last 元数据）；_release_staging_pool() 每次发送后归还 cupy 暂存池块并打 warning 日志（种子全量同步会经池传输最高 2× bucket_size 数据，此前对 torch 分配器不可见）；空 delta 走 slot 缓存零计数短回路（Opt-B，commit 4358b94a）。
5. 接收侧验证与测试配套：sglang_rollout/delta_loader.py 新增 _verify_dense()——用 snap_then_copy_ 包装 torch.Tensor.copy_，在真实加载路径执行前快照所有写目标，按最终状态逐位比对（而非逐次 copy_，规避 Mamba 多阶段 loader 的 raw-vs-transformed 误报）。测试分三层：TP>1 差分 oracle（test_mcore_probe_differential.py，真实 megatron_to_hf vs probe 组装位级一致，覆盖 qwen2 / qwen3_moe / nemotron_h / falcon_h1）、真 sglang 引擎闭环（test_sglang_delta_loop.py，扰动 → 稀疏 flush → update_weights_from_tensor → 幂等性验证）、CPU 验证扫描（test_sglang_loader.py 一致状态通过 / 分歧状态响亮失败两个用例）。

关键文件：

- verl/workers/engine/megatron/delta_export.py（模块 导出器；类别 source；类型 core-logic；符号 _ProbeGroup, _nan_block, _warm_lazy_mappings, make_probe）：新增 307 行的核心文件：基于 AutoBridge.get_conversion_tasks 构建导出索引，实现 _ProbeGroup 与 NaN 哨兵探针机制，是本次变更的心脏。
- verl/checkpoint_engine/delta_checkpoint_engine.py（模块 检查点引擎；类别 source；类型 core-logic；符号 _publish_values_flush, _release_staging_pool, init, _verify_due）：发送端核心逻辑：新增 verify_every 幂等性扫描、_publish_values_flush 的 verify 元数据、_release_staging_pool 回收 cupy 暂存池，是生产级验证链路的发送半场。
- verl/workers/engine/megatron/transformer_impl.py（模块 模型引擎；类别 source；类型 core-logic；符号 _mcore_export_index, get_per_tensor_param_shard, _hf_delta_entry, get_per_tensor_param_delta_shard）：MegatronEngine 的 delta 导出入口：_mcore_export_index 懒构建索引、get_per_tensor_param_shard / _hf_delta_entry / get_per_tensor_param_delta_shard 构成 mcore 侧协议实现，delta_pin_snapshots = False 是 30B/235B 内存策略的关键决策。
- verl/workers/rollout/sglang_rollout/delta_loader.py（模块 权重加载；类别 source；类型 core-logic；符号 verify_dense, snap_then_copy）：接收侧验证：verify_dense 以幂等性判定替代逐次copy比对，正确兼容sglang多阶段变换loader（Mamba~~A~~=-exp(A_log)）。

- tests/special_distributed/test_mcore_probe_differential.py（模块 差分测试；类别 test；类型 test-coverage；符号 _build_tiny_hf_dir, main）：TP>1 差分 oracle：真实 megatron_to_hf 与 probe 组装结果位级比对，覆盖 qwen2 / qwen3_moe / nemotron_h / falcon_h1，是 probe 两个核心假设的回归防线。
- tests/special_distributed/test_sglang_delta_loop.py（模块 闭环测试；类别 test；类型 test-coverage；符号 _build_tiny_hf_dir, _sparse_flush, _verify_flush, main）：真 sglang 引擎闭环测试：随机化 → 扰动 → 稀疏 flush → update_weights_from_tensor → 幂等性验证零失配，覆盖名字重映射、融合张量切分与多阶段 loader。
- tests/checkpoint_engine/test_sglang_loader.py（模块 加载器测试；类别 test；类型 test-coverage；符号 _dense_verify_flush, test_verify_sweep_passes_on_identical_state, test_verify_sweep_fails_loud_on_divergence）：CPU 上的验证扫描单测：一致状态通过、分歧状态响亮失败两个用例，是 _verify_dense 逻辑的低成本回归。
- verl/workers/engine/utils.py（模块 引擎工具；类别 source；类型 core-logic；符号 prime_delta_snapshots）：prime_delta_snapshots 签名调整：显式 pin 参数替代环境变量，是 review 驱动的策略归属收敛。
- verl/workers/engine/base.py（模块 引擎基类；类别 source；类型 configuration）：BaseEngine 解析类级 delta_pin_snapshots 默认值，统一各后端的快照 pin 策略入口。
- verl/utils/megatron_utils.py（模块 工具函数；类别 source；类型 core-logic）：make_megatron_module 容忍 NoPE 配置：NemotronH 系混合架构无 rope，缺失 rope_theta 在 bridge 路径上不再是错误。
- verl/workers/engine/spec.py（模块 协议定义；类别 source；类型 data-contract）：ShardSpec 契约的少量调整，保持与 #7144/#7085 分片描述协议的兼容。
- tests/checkpoint_engine/test_sharded_delta.py（模块 分片测试；类别 test；类型 test-coverage）：既有分片 delta 测试的配套微调，保持与 spec 变更同步。

关键符号：build_export_index, make_probe, mcore_export_index, get_per_tensor_param_shard, _hf_delta_entry, get_per_tensor_param_delta_shard, _verify_dense, snap_then_copy, _release_staging_pool, _publish_values_flush, prime_delta_snapshots

关键源码片段

verl/workers/engine/megatron/delta_export.py

新增 307 行的核心文件：基于 AutoBridge.get_conversion_tasks 构建导出索引，实现 _ProbeGroup 与 NaN 哨兵探针机制，是本次变更的心脏。

```
# verl/workers/engine/megatron/delta_export.py
# 探针核心：在 probe 副本上执行 mapping 真实的 megatron_to_hf 转换代码，
# 但把"通信"替换为本地合成，把"规模 / 秩"保留为真实值。
# 进程组在此携带两种可分离语义：size/rank（数值计算消费，例如 Mamba 的
# local_dim = global // tp_size）与 communication（真实集合通信）。
# 探针只替换后者：组变为 _ProbeGroup，bridge 的通信助手被 stub 成
# 本地合成——gather_from_tp_ranks 在本 rank 的真实下标处返回本分片，
# 其余 rank 位置用 NaN 占位（那些 rank 的贡献由它们自己的探针导出）。
```

```
class _ProbeGroup:
    """size/rank 保真的进程组替身：megatron_to_hf 内的数值计算读到真实
    并行规模；任何真正尝试通过该组通信的行为都会立刻抛错——探针已 stub
    掉 bridge 的通信助手，其他路径触达进程组即代表未覆盖的通信模式。"""

    def __init__(self, size: int, rank: int):
        self._size = int(size)
        self._rank = int(rank)

    def size(self) -> int:
        return self._size

    def rank(self) -> int:
        return self._rank

    def __getattr__(self, name):
        raise RuntimeError(
            f"probe process group asked for {name!r}: this mapping communicates "
            "outside the stubbed helpers (gather_from_tp_ranks / "
            "gather_from_ep_ranks[_scale] / pp broadcasts) -- extend "
            "make_probe's comm stubs before trusting its export"
        )


_NAN_POOL: dict = {}


def _nan_block(shape, dtype, device) -> torch.Tensor:
    """按 (shape, dtype, device) 池化的只读全 NaN 占位块：gather stub
    每次探针调用都为每个参数分发这些块，而它们只被读取（转换是函数式），
    因此每个不同 shape 只需一个块服务整个模型整个运行周期，避免每次
    同步都 cudaMalloc + fill（这在 235B 规模上是可测的稳态回归）。"""
    key = (tuple(shape), dtype, str(device))
    t = _NAN_POOL.get(key)
    if t is None:
        t = torch.full(tuple(shape), float("nan"), dtype=dtype, device=device)
        _NAN_POOL[key] = t
    return t
```

verl/checkpoint_engine/delta_checkpoint_engine.py

发送端核心逻辑：新增 verify_every 幂等性扫描、_publish_values_flush 的 verify 元数据、_release_staging_pool 回收 cupy 暂存池，是生产级验证链路的发送半场。

```
# verl/checkpoint_engine/delta_checkpoint_engine.py
# 种子同步走 values-only 全量线：wire 编码标签维持 "dense" —— 它是协议，
# 与接收侧 delta_loader 的解码共享。verify 标志使该 flush 同时充当
# 幂等性扫描的载荷：接收端把 trainer 的完整当前权重重放到已同步服务器，
# 必须是无操作。
```

```
def _publish_values_flush(
    self, params: list[DeltaParam], values: torch.Tensor, is_last: bool, verify: bool = False
) -> None:
    values = values.contiguous()
    empty_pos = torch.empty(0, dtype=torch.uint8, device=values.device)
    meta = {
        "is_full": True,
        "encoding": "dense",
        "is_last": is_last,
        "terminal_empty": False,
        "pos_numel": 0,
        "val_numel": int(values.numel()),
        "val_dtype": str(values.dtype).replace("torch.", ""),
        "spec": {
            "encoding": "dense",
            "verify": verify,
            "is_last": is_last,
            "params": [vars(p) for p in params],
            "checksum": int(_checksum(empty_pos, values)),
        },
    }
    self.socket.send_string(self.topic, flags=zmq.SNDMORE)
    self.socket.send_pyobj(meta)
    val_u8 = values.contiguous().view(torch.uint8)
    # 先落入 cupy 缓冲再广播：ray 的 NCCL broadcast 在独立流上入队，
    # 对零拷贝视图广播会与释放后的 allocator 复用竞争。
    val_cp = cp.empty(val_u8.numel(), dtype=cp.uint8)
    val_cp[:] = cp.asarray(val_u8)
    collective.broadcast(val_cp, src_rank=0, group_name=self.group_name)


def _release_staging_pool(self, phase: str) -> None:
    """把 cupy 暂存池的块归还 CUDA 并记录证据：种子全量同步经池传输最高
    2× bucket_size 数据，这些内存在归还前对 torch 分配器与裸 cudaMalloc
    （NCCL workspace、优化器缓冲）不可见，在紧张的 mcore 显存下会 OOM。"""
    from verl.utils.device import get_torch_device

    pool = cp.get_default_memory_pool()
    held = pool.total_bytes()
    free_before, _ = get_torch_device().mem_get_info()
    pool.free_all_blocks()
    free_after, _ = get_torch_device().mem_get_info()
    logger.warning(
        "cupy staging pool after %s send: held %.2fGB; device free %.2f->%.2fGB on release",
        phase, held / (1 << 30), free_before / (1 << 30), free_after / (1 << 30),
    )
```

verl/workers/engine/megatron/transformer_impl.py

MegatronEngine 的 delta 导出入口：_mcore_export_index 懒构建索引、get_per_tensor_param_shard / _hf_delta_entry / get_per_tensor_param_delta_shard 构成 mcore 侧协议实现，delta_pin_snapshots = False 是 30B/235B 内存策略的关键决策。

```
# verl/workers/engine/megatron/transformer_impl.py（MegatronEngine）
# mcore 常驻模型并行本地参数并每步搬运大型 host 缓冲区；在其上再整组
# pin 分片快照会耗尽节点 cudaHostAlloc 池，表现为无关分配的 CUDA OOM
# （30B/235B 规模实测）——故 mcore 默认 pageable 主机内存。
delta_pin_snapshots = False


def _mcore_export_index(self):
    """一次性构建逐参数 delta 导出索引：几何 spec 与 form-B 探针均由
    bridge 的转换任务推导（见 delta_export.build_export_index）。
    delta_sharded 基于 Megatron-Bridge 参数映射，vanilla mbridge 与
    LoRA 在此直接断言拒绝。"""
    index = getattr(self, "_delta_export_index", None)
    if index is None:
        from .delta_export import build_export_index

        assert not self.vanilla_bridge, (
            "megatron delta_sharded is built on Megatron-Bridge param "
            "mappings; the deprecated vanilla mbridge flavor is not supported"
        )
        assert self.peft_cls is None, "megatron delta_sharded does not support LoRA"
        index = build_export_index(self.bridge, self.module)
        self._delta_export_index = index
        self._delta_export_by_name = {rec.megatron_name: rec for rec in index}
        self._delta_slot_cache = {}
    return index


def get_per_tensor_param_shard(self, **kwargs):
    """逐个产出每个 rank 的本地 mcore 分片 (name, local_flat_bf16, ShardSpec)：
    spec 只携带线上合并组（几何全部由通信 stub 的探针负责），
    纯导出无副作用；TP + EP 仅限（PP=1 在索引构建处断言）。"""
    load_megatron_model_to_gpu(self.module, load_grad=False)
    index = self._mcore_export_index()

    def _gen():
        for rec in index:
            local = rec.param.data
            if local.is_floating_point() and local.dtype != torch.bfloat16:
                local = local.to(torch.bfloat16)
            yield rec.megatron_name, local.reshape(-1), rec.spec

    return _gen(), None

def _hf_delta_entry(self, name, spec, place, lidx, lval):
```

```
    """mcore 的逐参数 entry 构建：用 NaN 哨兵探针调用 bridge 自己的转换器
    （通信被 stub 的副本——真实组规模、本地合成 gather），
    槽位表冻结在 slot_cache 后，空 delta 可走零计数快路径。"""
    from .delta_export import mcore_hf_delta_entry

    rec = self._delta_export_by_name[name]
    return mcore_hf_delta_entry(rec, place, lidx, lval, self._delta_slot_cache)
```

评论区精华

Issue 评论区有两轮关键问答，均发生在 #7060 追踪 issue 下：

1. ISEEKYAN 询问是否有定义 delta 格式的协议文件，倾向训练后端无关协议（类似 HF safetensor 之于全量参数）；ChangyiYang 回应每个训练后端实现 get_per_tensor_param_delta_shard，其输出格式即协议——最终 HF 坐标稀疏 entry，定义在 verl/workers/engine/spec.py，线上编码在 delta_checkpoint_engine.py 头部；越过该边界后的 gather、分桶、接收端全部后端无关。
2. wuxibin89 询问是否支持流水线并行；ChangyiYang 明确 PP 不支持且会立即抛错，由 @gxlvera 后续支持。

另有 commit 0b3e9745 记录的 review 反馈：策略旋钮还给所有者（policy knobs move to their owners）——快照 pin 从环境变量 VERL_DELTA_PIN 改为 BaseEngine 类级 delta_pin_snapshots 属性，由 MegatronEngine 覆盖为 False。

- delta 格式协议文件与后端无关性 (design): 协议边界确认：spec.py 的 ShardSpec 与 HF 坐标 entry 为后端无关协议，mcore 实现遵循同一契约。
- 流水线并行（PP）支持范围 (question): PP 明确超出本 PR 范围，导入器显式报错兜底，后续由 @gxlvera 承接。
- 快照 pin 策略的归属（review 驱动调整） (design): 内存策略归引擎所有者决策，mcore 默认 pageable 快照。

风险与影响

- 风险：
  1. 适用范围硬边界：PP/VPP>1 与 LoRA 在 _mcore_export_index 与索引构建处断言报错，使用这些配置的 Megatron 用户无法启用 delta_sharded，需要显式 fallback 到全量同步路径，文档中应明确提示。
  2. probe 假设依赖手工回归测试：两个核心假设（通信限定在 stub 助手内、变换重排而非算数混合——混合会吃掉 NaN 哨兵）的守护者是 test_mcore_probe_differential.py，它是手工驱动的 special_distributed 脚本、未接入 CI；Megatron-Bridge 升级后若未重跑可能静默失真。
  3. 测试脚本残留个人路径：test_sglang_delta_loop.py 与 test_mcore_probe_differential.py 中 sys.path.insert(0, os.environ.get("VERL_PATH", "/home/changyi/verl_ab_pre")) 的缺省值指向作者本机目录，他人在无 VERL_PATH 环境变量时运行可能导入错误代码。

4. 全局 monkey-patch：_verify_dense 临时替换 torch.Tensor.copy_ 是进程级操作，仅在显式开启 verify_every 时执行，但扫描期间其他线程的并发 copy_ 会被纳入快照统计。
5. 内存权衡：cupy 暂存池在种子同步后会持有最多 2× bucket_size 内存（已通过 _release_staging_pool 回收并打日志）；快照默认 pageable 以规避 cudaHostAlloc 池饥饿（30B/235B 下实测），代价是每步快照读写的性能损失。 - 影响：对用户的直接收益：Megatron-Bridge 用户（TP + EP 配置）可获得 8.2–9.7× 的权重同步加速（235B 稳态 23.6 s vs 全量 NCCL 235.0–237.6 s），并继承 seed 全量导出的断点续训能力、sglang 侧无全模型镜像的内存画像。对系统而言，这是 delta 协议第三个后端（FSDP / veomni / mcore）的落地，验证了「输出格式即协议」的后端无关设计，为 #7060 路线图的 PP 支持及后续优化（closed-form 索引表 #7060、PP 稳态中继）铺路。对团队而言，23 个 commit 呈现高频自审与迭代过程，是后续后端接入的参考范本。 - 风险标记：仅限 TP + EP（PP 立即报错），probe 假设依赖手工回归测试，快照默认 pageable 内存，全局 patch copy_ 的验证扫描，测试脚本残留个人路径

关联脉络

- PR #7085 [veomni] feat: EP-aware sharded delta export (fused expert stacks): 同一条 delta_sharded 功能线：本 PR rebase 到含 #7085 的最新 main，diff 仅剩 megatron/engine 增量；共享 ShardSpec / BlockPlacement 契约并复用 tests/checkpoint_engine/test_sharded_delta.py 的验证思路。
- PR #7161 [fsdp] refactor: move unfuse_moe_params to FSDP backend: 同属权重同步 /checkpoint 引擎改造线：将 MoE 参数还原逻辑移入 FSDP 后端，与 delta_sharded 的 shard 导出体系互补，二者共同演进 checkpoint 引擎的权重传输路径。
- PR #7205 [ckpt] feat: add hccl ckpt engine split_weight_chunks: 同属 checkpoint_engine 家族：在 NPU 侧探索分块权重传输，与 delta 协议共享「分片描述 + 稀疏传输」的设计方向。