PR #7085 完整报告

verl-project/verl

[veomni] feat: EP-aware sharded delta export (fused expert stacks)

合并时间：2026-07-29 11:10

原文链接：http://prhub.com.cn/verl-project/verl/pull/7085

执行摘要

- 一句话：VeOmni EP 感知分片 delta 导出，大幅加速 MoE 权重同步
- 推荐动作：值得仔细阅读。特别关注：(1) 如何通过 handler 自身探测 slot 表（enumerate_hf_slots）来避免手动维护 slot 表；(2) NaN sentinel 的设计使得每个 rank 可以独立完成 delta 转换，无需 rank 0 全量重建；(3) BlockPlacement 如何统一表达 EP 手动分割和 FSDP DTensor 分割。

功能与动机

VeOmni 后端使用融合专家堆栈（如 gate_up_proj）并结合 EP+FSDP2 混合并行，原有的分片 delta 引擎无法正确处理其 shard geometry，导致无法使用 delta_sharded 模式进行高效权重同步。本 PR 填补了这一空白，使得 VeOmni 用户也能享受分片 delta 带来的通信节省。PR body 明确指出这是 review-requested split 的第二部分（引擎核心 +FSDP 已移至 #7144），在本 PR 中“wires veomni FSDP2+EP into the sharded delta sync end to end”。

实现拆解

1. 重构 MoE 参数转换器：将 _map_moe_params_common 和 default_moe_param_handler 的 ep_rank 参数改为 expert_id_base，使其支持任意起点索引，从而兼容 EP 局部快照和全局全量。
2. 新增 VeOmni 专用 converter machinery：在 verl/workers/engine/veomni/utils.py 中实现 slot 表自动枚举（enumerate_hf_slots）、单行转换（convert_row_to_hf）、delta entry 构建器（hf_entry_converter）以及顶层导出生成器（veomni_shard_export），支持融合专家堆栈到 HF 独立命名的映射。
3. 扩展 spec 层：在 ShardSpec 中新增 contributes 字段，用于处理 HSDP 副本维度；将 derive_placement 重命名为 derive_dtensor_placement 并明确其职责；同时修改 hf_delta_export，使其在 spec.place 已显式设置时直接使用 exporter 提供的 placement，而不经过 DTensor 推导。
4. 在 VeOmni 引擎中接入：替换 transformer_impl.py 中 get_per_tensor_param_shard 的 NotImplementedError 为实际实现，通过调用 veomni_shard_export 获取带 Spec 的 shard，并实现 _hf_delta_entry 方法，对含 converter 的 spec 分派到 hf_entry_converter，否则回退到父类的 DTensor 逻辑。

5. 配套测试与文档：在 test_sharded_delta.py 中新增 test_hf_delta_export_converter_param 等测试，覆盖 converter 参数的 delta 导出、非平凡 block placement、以及 HSDP replica rank 锁步行为；更新 delta_weight_sync.md 文档。

关键文件：

- verl/workers/engine/veomni/utils.py（模块 VeOmni 引擎；类别 source；类型 core-logic；符号 _map_moe_params_common, default_moe_param_handler, enumerate_hf_slots, convert_row_to_hf）：核心逻辑所在，新增所有 converter machinery：slot 表枚举、NaN 探针、delta entry 构建、顶层导出生成器。变更量最大（+256/-7）。
- verl/workers/engine/veomni/transformer_impl.py（模块 VeOmni 引擎；类别 source；类型 dependency-wiring；符号 _with_offload_back, _hf_delta_entry）：引擎入口集成，将 get_per_tensor_param_shard 从 NotImplementedError 改为实际实现，新增 _hf_delta_entry 进行 converter dispatch。
- tests/checkpoint_engine/test_sharded_delta.py（模块 分片 delta；类别 test；类型 test-coverage；符号 test_hf_delta_export_converter_param, test_hf_delta_export_converter_nontrivial_block, test_hf_delta_export_replica_rank_stays_lockstep, test_derive_dtensor_placement_unsharded）：新增测试覆盖 converter 参数 delta 导出、非平凡 block placement、HSDP replica 锁步等场景。
- verl/workers/engine/spec.py（模块 规格层；类别 source；类型 core-logic；符号 derive_dtensor_placement, ShardSpec.contributes）：基础 spec 扩展：新增 contributes 字段，重命名 derive_placement 为 derive_dtensor_placement，调整 docstring。
- verl/workers/engine/utils.py（模块 引擎工具；类别 source；类型 dependency-wiring）：hf_delta_export dispatch 修改：当 spec.place 已设置时直接使用 exporter 提供的 placement，而非通过 derive_dtensor_placement 推导。
- docs/advance/delta_weight_sync.md（模块 文档；类别 docs；类型 documentation）：更新文档以反映 VeOmni 支持。

关键符号：_map_moe_params_common, default_moe_param_handler, enumerate_hf_slots, convert_row_to_hf, hf_entry_converter, veomni_shard_export, _is_ep_param, get_per_tensor_param_shard, _hf_delta_entry, derive_dtensor_placement, hf_delta_export

关键源码片段

verl/workers/engine/veomni/transformer_impl.py

引擎入口集成，将 get_per_tensor_param_shard 从 NotImplementedError 改为实际实现，新增 _hf_delta_entry 进行 converter dispatch。

```
def get_per_tensor_param_shard(self, **kwargs):
    """Yield each rank's *local* shard with its ShardSpec."""
```

```
    from .utils import veomni_shard_export

    # 判断是否需要手动 offload（CPUOffloadPolicy 下不需要）
    manual_offload = not getattr(self, "_uses_fsdp2_cpu_offload_policy", False)
    if manual_offload:
        load_veomni_model_to_gpu(self.module)
    # 调用专用导出函数获取生成器和元数据
    gen, meta = veomni_shard_export(self.module)

    def _with_offload_back():
        """包装生成器，在迭代结束后执行 offload（如果适用）"""
        yield from gen
        if manual_offload and self._is_offload_param:
            offload_veomni_model_to_cpu(self.module)

    return _with_offload_back(), meta


def _hf_delta_entry(self, name, spec, place, lidx, lval):
    """veomni 的每参数 delta entry 构建：对含 converter 的 fused expert 参数使用
    hf_entry_converter，否则回退到父类的 DTensor 处理。"""
    from ..spec import BlockPlacement
    from .utils import NO_SLOTS_MSG, hf_entry_converter

    # 如果 spec 携带了完整的 converter 信息（to_hf_chunk + hf_slots），
    # 并且 place 是 BlockPlacement，则使用 hf_entry_converter 构建 entry
    if spec.to_hf_chunk is not None and isinstance(place, BlockPlacement) and spec.hf_slots is
    not None:
        return hf_entry_converter(name, spec, place, lidx, lval)
    # 如果只有 to_hf_chunk 但没有 hf_slots，则无法在发送端完成转换，报错
    if spec.to_hf_chunk is not None:
        raise NotImplementedError(f"{name}: {NO_SLOTS_MSG}")
    # 其余参数走父类的 DTensor identity 处理
    return super()._hf_delta_entry(name, spec, place, lidx, lval)
```

评论区精华

- wuxibin89 要求将引擎核心重构与 VeOmni 集成分离为两个 PR（#7144 和本 PR），作者照做。
- gemini-code-assist[bot] 指出 gather_dense_blocks_to_rank0 中 dist.gather 的 dst 参数需使用 group-relative rank 0，而非 dist.get_global_rank。
- wuxibin89 担心全量 materialize 专家内存消耗（如 kimi-k2.5 的 gate_up_proj 21GB），作者澄清该实现使用分片方式避免 materialize。
- wuxibin89 询问 snapshot 是否仅在第一部 seed，作者解释 hf_delta_export 在每次 sync 时都会刷新 snapshot（diff 后 snap.copy_(local)），无需额外 seed。
- wuxibin89 建议将 get_per_tensor_param_shard 实现移至 utils.py 以保持简洁，作者在后续 commit 中重构。

- PR 拆分请求 (design): 作者将引擎核心 +FSDP 部分移至 #7144，本 PR 仅包含 VeOmni EP 导出（叠加在 #7144 之上）。
- dist.gather 目标 rank 应为 group-relative 而非 global (correctness): 接受建议，修改为 group-relative rank 0。
- 全量 materialize 专家内存风险 (performance): 作者澄清该实现使用分片方式，每个 rank 仅持有自己的 block，不会 materialize 全量。
- Snapshot 刷新时机 (correctness): 作者解释 hf_delta_export 在每次 sync 时都会刷新 snapshot（diff 后 snap.copy_(local)），所以每个 sync 后 snapshot 都对应最新状态，无需额外 seed。
- get_per_tensor_param_shard 实现位置 (style): 作者采纳建议，在 50b488bd 中将 DTensor+EP 声明逻辑移至 utils.py 的 veomni_shard_export。

风险与影响

- 风险：依赖风险：本 PR 基于 #7144 的引擎核心重构，若 #7144 有回归会影响本 PR。 正确性风险：converter machinery 涉及 EP+FSDP2 混合并行 placement 推导，若 BlockPlacement 计算错误可能导致权重损坏或训练发散。 性能风险：每次 sync 中增加的 converter 调用（每参数 NaN probe 和 slot lookup）可能引入额外开销，但从 benchmark 看收益远超成本。 兼容性风险：hf_delta_export 的 dispatch 修改可能影响其他后端（如纯 FSDP），需确保回归测试覆盖。 内存风险：虽然避免全量 materialize，但 NaN probe 需要构造临时 tensor，对于超大 MoE 仍应注意上限。
- 影响：用户影响：VeOmni 后端用户现在可使用 delta_sharded 模式，获得 4-21 倍的权重同步加速。 系统影响：分片 delta 引擎的能力从仅支持 FSDP 扩展到支持 EP+FSDP 混合并行，为后续支持其他后端（如 mcore）奠定基础。 团队影响：完成了重构路线图中的重要一步，使 VeOmni 能充分利用 delta 增益。
- 风险标记：依赖上游重构 (#7144), EP+FSDP 混合并行 placement 复杂，新增 converter 可能引入回归，NaN Probe 临时内存开销

关联脉络

- PR #7144 [ckpt, fsdp] feat: sharded delta block placements + backend-owned HF export (engine core + FSDP): 本 PR 基于 #7144 的引擎核心 +FSDP 重构，是本 PR 的前置依赖。
- PR #7080 [veomni] EP-aware sharded delta export (fused expert stacks): 被本 PR 取代的更早实现（已关闭）。
- PR #6612 [veomni] full-weight ep export: 讨论中提及的全量权重 EP 导出方案，本 PR 通过 delta 方法彻底优化了该路径。
- PR #7060 issue/PR about slot table mechanism: 在 NO_SLOTS_MSG 中引用 (#7060)，可能是 slot 表机制的原始 issue 或 PR。