PR #1806 完整报告

THUDM/slime

feat: delta weight sync (disk + nccl transports)

合并时间：2026-05-26 11:52

原文链接：http://prhub.com.cn/THUDM/slime/pull/1806

执行摘要

- 一句话：引入 delta 权重同步，支持 disk 和 NCCL 传输
- 推荐动作：值得精读的设计型 PR。模板方法模式和编码器 decoupling 清晰，wire protocol 设计兼顾跨 DC 场景。建议关注 _bytewise_diff_mask 和 _sparse_boundaries 的高效实现（单次 concat + nonzero + searchsorted），以及 _flush_bucket 中 nccl/disk 分支的对称逻辑。

功能与动机

核心动机是训练 / 推理分离（cross-DC disaggregation）—— trainer 和 rollout engine 位于不同数据中心，共享文件系统带宽仅数百 MB/s，全量广播不可行，但稀疏 delta（约 3% 密度，355B 模型约 5 GB）可行。受 arXiv:2509.19128（选择性覆盖）和 Fireworks AI 的 cross-DC RL 成本分析启发。

实现拆解

1. 基类模板方法化：UpdateWeightFromDistributed 中将 update_weights 的循环迭代提取为 _send_weights(pbar)，新增 _on_chunk 钩子。基类保留 full 模式的原有逻辑，delta 子类通过覆盖这两个方法注入稀疏 diff + 编码流程。新增 pop_metrics 供 actor 收集同步性能指标。
2. Delta 同步实现：新增 UpdateWeightFromDistributedDelta（update_weight_from_distributed_delta.py）。其 _send_weights 维护 pinned-CPU 快照，逐参数调用 _bytewise_diff_mask 得到 bool mask，经 _sparse_boundaries 合并非零位置后，按 --update-weight-encoding（indices/deltas/deltas_zstd）编码位置。每 bucket 调用 _flush_bucket：nccl 直接 broadcast positions+values，disk 写入 safetensors 文件。同步结束时 _finalize_sync 处理 HTTP push（disk 模式）。
3. SGLang 接收端支持：修改 sglang_engine.py 中 update_weights_from_distributed 和 update_weights_from_disk，新增 load_format='delta' 和 delta 参数（序列化为 JSON，携带 DeltaSpec）。sglang patch 实现 _apply_delta_payload，通过 _delta_apply_context 劫持 Tensor.copy_ 实现 NaN 掩码覆盖。
4. 参数配置与校验：arguments.py 新增 6 个参数（--update-weight-mode, --update-weight-transport, --update-weight-encoding, --update-weight-delta-dir, --update-weight-delta-keep-files, --custom-delta-pre-push-path），并校验 delta 模式不能与 --colocate 同时使用。

5. 示例与文档：*examples/delta_weight_sync/run-glm4.7-355B-A32B-delta.sh* 演示 355B 模型配置；*docs/en/advanced/delta-weight-sync.md* 详细说明原理、参数和部署建议。

关键文件：

- slime/backends/megatron_utils/update_weight/update_weight_from_distributed_delta.py（模块 权重同步；类别 source；类型 core-logic；符号 ParamDiff, EncodedChunk, empty, _checksum）：新增 delta 权重同步核心实现，包含 diff 计算、编码、传输分支，是 PR 的主文件。
- slime/backends/megatron_utils/update_weight/update_weight_from_distributed.py（模块 权重同步；类别 source；类型 core-logic；符号 pop_metrics, _send_weights, _on_chunk, _iter_non_expert_chunks）：重构基类，提取模板方法 _send_weights/_on_chunk，新增 pop_metrics；影响所有同步路径。
- slime/backends/sglang_utils/sglang_engine.py（模块 引擎接口；类别 source；类型 core-logic；符号 update_weights_from_disk, update_weights_from_distributed）：SGLang 引擎侧新增 delta 参数传递和 disk/NCCL 入口，是接收端协议适配的关键。
- slime/utils/arguments.py（模块 参数配置；类别 source；类型 configuration）：新增 6 个 CLI 参数控制 delta 模式、传输、编码等，并添加参数校验逻辑。
- docker/patch/latest/sglang.patch（模块 SGLang 补丁；类别 infra；类型 core-logic；符号 UpdateWeightFromDiskReqInput, UpdateWeightsFromDistributedReqInput, update_weights_from_tensor）：sglang 补丁包含接收端 _apply_delta_payload 实现、DeltaSpec/DeltaParam 定义，是完整 delta 链路的另一半。
- docs/en/advanced/delta-weight-sync.md（模块 文档；类别 docs；类型 documentation）：详细文档，帮助用户理解 delta 同步原理、参数含义和部署注意事项。

关键符号：UpdateWeightFromDistributed._send_weights, UpdateWeightFromDistributed.pop_metrics, UpdateWeightFromDistributedDelta._bytewise_diff_mask, UpdateWeightFromDistributedDelta._sparse_boundaries, UpdateWeightFromDistributedDelta.encode_indices, UpdateWeightFromDistributedDelta.encode_deltas, UpdateWeightFromDistributedDelta._flush_bucket, SGLangEngine.update_weights_from_distributed, SGLangEngine.update_weights_from_disk

关键源码片段

slime/backends/megatron_utils/update_weight/update_weight_from_distributed_delta.py

新增 delta 权重同步核心实现，包含 diff 计算、编码、传输分支，是 PR 的主文件。

```
# _bytewise_diff_mask：byte-level diff 检测 —— view-as-integer 避免 dtype 特化
# current, snapshot：同一 shape 的完整参数张量，snapshot 是上一次 broadcast 的 CPU pinned 副本
def _bytewise_diff_mask(current: torch.Tensor, snapshot: torch.Tensor) -> torch.Tensor:
    """
    逐元素 bool mask：True 表示 current 与 snapshot 在该位置的 bytes 不同。
```

```
    通过 view-as-integer 实现，无 dtype 特化，支持 bf16/fp16/...
    """
    es = current.element_size()
    # 根据元素字节大小选择对应的整数 dtype
    int_dtype = {1: torch.uint8, 2: torch.int16, 4: torch.int32, 8: torch.int64}.get(es)
    if int_dtype is None:
        raise ValueError(f"unsupported element size {es}")
    return current.view(int_dtype) != snapshot.view(int_dtype)


# _sparse_boundaries：一次 concat + nonzero + searchsorted 获取每个参数在合并结果中的边界
def _sparse_boundaries(
    diffs: list[ParamDiff],
) -> tuple[torch.Tensor, list[int], torch.Tensor, list[int]]:
    """
    将所有 ParamDiff 的 values 和 masks 拼接，然后一次 nonzero 得到全局非零索引，
    最后通过 searchsorted 切分到各参数。
    将 O(num_params) 次 host sync 降为 O(1)。
    """
    device = diffs[0].values.device
    sizes = [d.values.numel() for d in diffs]
    cum = list(itertools.accumulate(sizes))
    cum_t = torch.tensor(cum, dtype=torch.int64, device=device)

    big_values = torch.cat([d.values.contiguous().view(-1) for d in diffs], dim=0)
    big_mask = torch.cat([d.mask.contiguous().view(-1) for d in diffs], dim=0)
    big_idx = big_mask.nonzero(as_tuple=False).view(-1)
    big_val = big_values[big_idx]
    # 每个参数在此 chunks 内的结束位置（元素级）
    bounds = torch.searchsorted(big_idx, cum_t).tolist()
    return big_val, bounds, big_idx, cum

# encode_indices：使用 int32 绝对位置编码（最直接，体积最大，计算最少）
def encode_indices(param_name: str, local_positions: torch.Tensor, local_values: torch.Tensor) -
> EncodedChunk:
    """
    将每个参数内的局部非零位置编码为 int32 绝对位置列表。
    local_positions：1-D int64，相对该参数 flatten 后的偏移。
    local_values  ：1-D，对应位置的值。
    """
    pos_np = local_positions.cpu().numpy().astype(np.int32)
    val_tensor = local_values.contiguous()
    nnz = pos_np.shape[0]
    # 构建一个 DeltaParam 描述
    param = DeltaParam(
        name=param_name,
        length=len(pos_np),
        offset_bytes=0, # 实际 offset 在合并时确定
        pos_bytes_len=pos_np.nbytes,
```

```
        encoding="indices",
    )
    return EncodedChunk(
        pos_bytes=pos_np.tobytes(),
        val_tensor=val_tensor,
        params=[param],
        nnz=nnz,
    )
```

slime/backends/megatron_utils/update_weight/update_weight_from_distributed.py

重构基类，提取模板方法_send_weights/_on_chunk，新增pop_metrics；影响所有同步路径。

```
class UpdateWeightFromDistributed:
    """
    更新分布式引擎权重的基类。
    update_weights 调用 _send_weights(self, pbar)，子类通过覆盖 _send_weights 和 _on_chunk
    实现不同策略。
    """

    def __init__(self, args, model, weights_getter, *, model_name, quantization_config):
        self.args = args
        self.model = model
        self.model_name = model_name
        self.quantization_config = quantization_config
        self.weight_version = 0
        self._model_update_groups = None
        self.update_weight_metrics: dict[str, float] = {} # 存储同步计时等指标

    def pop_metrics(self) -> dict[str, float]:
        """
        返回并清空 `update_weight_metrics`。actor 在 step 结束时 drain 并记录日志。
        """
        out, self.update_weight_metrics = self.update_weight_metrics, {}
        return out

    @torch.no_grad()
    def update_weights(self) -> None:
        self.weight_version += 1
        # ... (pause, flush, int4 pre-process) ...
        dist.barrier(group=get_gloo_group())

        pbar = tqdm(...) if self._is_pp_src_rank else None
        self._send_weights(pbar) # 子类实现负责遍历参数并发送

        # ... (int4 post-process, continue) ...

    def _send_weights(self, pbar):
        """
```

```
        默认实现：遍历所有非 expert 和 expert 参数，调用 `_on_chunk` 处理每批 bucket。
        delta 子类覆盖此方法，跳过直接迭代，改为 diff+编码+flush。
        """
        # full 模式原有逻辑移入此方法
        ...
```

评论区精华

审核期间无实质性技术讨论，核心维护者 zhuzilin 直接批准合并。PR body 和代码注释已完整陈述设计决策。

- 暂无高价值评论线程

风险与影响

- 风险：
  - 核心路径变更：重构 UpdateWeightFromDistributed 基类的 update_weights 方法，影响现有 full sync 流程，需要确保正确性。
  - 新的传输方式风险：disk 传输依赖共享文件系统（--update-weight-delta-dir），若文件系统延迟高、写入冲突或多引擎并发读取未严格同步，可能导致数据不一致或应用失败。
  - 无损性保证：receiver 使用 NaN 掩码覆盖保证无损，但需要验证 NaN 在 bfloat16/fp16 等 dtype 下的行为，防止意外传播。
  - 兼容性：--colocate 模式不支持 delta，需在参数校验层确保拒绝。
  - 性能：编码选择影响带宽 /CPU 权衡，deltas_zstd 压缩在 CPU 侧引入额外计算，可能影响同步延迟。
- 影响：
  - 用户：需要使用 --update-weight-mode=delta 并配合传输参数；disk 传输需要共享存储和 HTTP 推送配置。对跨 DC 训练用户影响显著——带宽需求从全量降至稀疏 delta（~3%）。
  - 系统：新增 UpdateWeightFromDistributedDelta 类，依赖 ray 和 safetensors；增量同步无额外同步开销（如 no sync barrier 冲突）。
  - 团队：需维护两个同步路径（full/delta）及 sglang patch；pop_metrics 提供了可观测性基座，便于未来优化。
  - 风险标记：核心路径变更，跨数据中心依赖，新传输协议，缺少端到端测试覆盖（此 PR 中），参数校验有限

关联脉络

- PR #1991 [ci] Add e2e test for delta weight update: 为 delta 同步添加端到端 CI 测试，验证此 PR 的完整链路。
- PR #1993 Patch sglang 0.5.12.post1 for delta sync: 更新 sglang 补丁以支持 delta 同步，与此 PR 的 patch 有重叠。
- PR #1975 [release] bump to v0.3.0: v0.3.0 发布包含此 PR 及后续相关 PR，作为版本集成的里程碑。