PR #2581 完整报告

radixark/miles

dashboard: resolve CP/PP-sharded train dumps offline and fix a partition-reader race

合并时间：2026-08-18 08:36

原文链接：http://prhub.com.cn/radixark/miles/pull/2581

执行摘要

- 一句话：修复 CP/PP 分片下 dashboard 视图错位与 --follow 双计数
- 推荐动作：值得精读。设计上把 split/assemble 收敛到同一个偏移函数并用 round-trip 测试固定，是保持并行语义一致的典范。旧 dump 的布局恢复通过"内容分组 + 宽度校验"避免了破坏性迁移，值得借鉴。并发 race 修复方式简单直接（锁住整个读 - 解析 - 追加事务），适合作为类似场景的参考。

功能与动机

四个用户可见 dashboard bug 都是在调试一台 GB300 16 节点 GLM-5.2 训练（TP2/CP4/PP4）时发现的：CP > 1 时 token 视图因拼接顺序错误而崩溃或错位；PP 下 batch 视图的 per-token 列为空；avg staleness 因前端重算公式差一而显示 2.75 而非 1.75；--follow 的 PartitionedReader._block 因 read-modify-write 横跨两个释放 GIL 的步骤，导致同一字节范围被重复追加，一个 lane 渲染出 147 个 segment 而真实只有 49 个。PR body 明确要"resolve CP/PP-sharded train dumps offline and fix a partition-reader race"。

实现拆解

1. dump 元数据增强：miles/utils/train_dump_utils.py 的 save_debug_train_data_for_rank 新增 cp_rank / cp_size / qkv_format 参数并写入 torch.save 字典；save_debug_train_data 从 get_parallel_state() 读取当前 CP 状态传入，测试与离线工具仍可显式传参，保持无 parallel state 可用。
2. CP 逆运算：miles/backends/training_utils/cp_utils.py 的 get_logits_and_tokens_offset_with_cp 增加可选 cp_rank / cp_size，默认仍走 get_parallel_state()；新增 assemble_log_prob_from_cp，用同一偏移函数将各 rank 切片写回完整 response，缺 rank 直接断言失败。
3. reader 重构：miles/dashboard/dump_reader.py 的 load_joined 改为先收集所有 shard 的 handles 与 (shard, row) 位置索引，再通过 _cp_layout（新 dump 用元数据，旧 dump 用 _recover_cp_layout_legacy 按 rollout_log_probs 内容分组加宽度校验）确定 cp_size / layout；_resolve_per_token_fields 对 _CP_SHARDED_FIELDS 逐字段选择持有者（PP 只有最后 stage 有）、处理 TP 副本去重、调用 assemble_log_prob_from_cp 重组，任何无法对齐的样本标记 alignment_failed。
4. 并发修复：miles/dashboard/store.py 的 _PartitionReader._block 用 threading.Lock 将 offset 读取、文件读取、解析、缓存更新变为一个事务。

5. 前端与测试配套：charts.js / views_tokens.js 支持 gap 断线和 bands 阴影，mask 位置序列化为 null；新增 test_cp_log_prob_assembly.py 和 test_partitions.py 并发用例；advisory.py 将 alignment_failed 呈现为 advisory。CI 上还移除了 CP 组装测试中多余的默认 register_cpu_ci 调用以通过测试守卫。

关键文件：

- miles/dashboard/dump_reader.py（模块 读取器；类别 source；类型 core-logic；符号 _cp_layout, _recover_cp_layout_legacy, _cp_widths_match, _build_train_row）：核心读取逻辑：在 load_joined 中按 shard 收集全部 handles 与位置索引，并通过 _cp_layout / _recover_cp_layout_legacy / _resolve_per_token_fields 按 PP/CP/TP 规则重组每个 per-token 字段到完整 response 长度，无法对齐的样本标记 alignment_failed。
- miles/backends/training_utils/cp_utils.py（模块 CP 工具；类别 source；类型 core-logic；符号 assemble_log_prob_from_cp, get_logits_and_tokens_offset_with_cp）：新增 assemble_log_prob_from_cp 作为 CP 切片的精确逆函数，并让 get_logits_and_tokens_offset_with_cp 支持显式 cp_rank/cp_size，保持 split/assemble 两侧共用同一偏移计算。
- miles/utils/train_dump_utils.py（模块 训练转储；类别 source；类型 data-contract；符号 save_debug_train_data, save_debug_train_data_for_rank）：dump 时记录 cp_rank/cp_size/qkv_format，使离线 reader 能确切知道每个 shard 的切片归属，是后续 CP 逆运算的前提。
- miles/dashboard/store.py（模块 存储层；类别 source；类型 core-logic；符号 _PartitionReader._block, _PartitionReader.init）：修复 PartitonedReader._block 的读 - 改 - 写 race，加锁使 offset 读取、文件读取、解析和追加成为原子事务，消除 --follow 双计数。
- miles/dashboard/static/charts.js（模块 图表；类别 source；类型 core-logic；符号 drawChart）：前端图表修复折线跨越 null gap 画直线的问题，并支持 bands 阴影提示 prompt/mask 区域，是 token 视图可读性的关键支撑。
- tests/fast/dashboard/test_partitions.py（模块 分区测试；类别 test；类型 test-coverage；符号 _counting_reader, test_concurrent_fill_does_not_duplicate_records, test_concurrent_refresh_does_not_duplicate_records）：新增两个确定性并发测试，覆盖 follow 线程与请求处理线程在同一个陈旧 offset 竞争的场景，验证不重复追加记录。
- tests/fast/backends/training_utils/test_cp_log_prob_assembly.py（模块 CP 测试；类别 test；类型 test-coverage；符号 _split, test_split_then_assemble_is_identity_thd, test_split_then_assemble_is_identity_bshd, test_partial_group_is_rejected）：用 round-trip 测试固定 CP split/assemble 逆运算，覆盖 thd/bshd 与缺失 rank 拒绝，是 CP 重组正确性的重要保障。
- miles/dashboard/static/views_tokens.js（模块 Token 视图；类别 source；类型 core-logic；符号 renderTokens, loadTokensPane）：token 视图补充 prompt/masked bands、null 序列化处理，并对缺失 conversation 给出提示，避免静默丢弃向用户误导。
- miles/dashboard/advisory.py（模块 告警；类别 source；类型 core-logic；符号 _rollout_advisories）：将 alignment_failed 样本转化为用户可见 advisory，区分“未 dump”和“dump 了但无法对齐”两种场景。

- tests/fast/dashboard/test_dump_reader_views.py（模块 视图测试；类别 test；类型 test-coverage；符号 test_tokens_null_the_stats_where_the_loss_is_masked）：调整 mask=0 位置的序列化断言为 null，防止占位零被误认为真实数值，与 chart 的 gap/bands 逻辑呼应。
- miles/dashboard/static/views_rollout.js（模块 Rollout 视图；类别 source；类型 core-logic）：调整 rollout 视图，避免前端自行重算 staleness，改为读取后端计算的 per-sample 值。
- miles/dashboard/serve.py（模块 服务入口；类别 source；类型 core-logic）：服务入口适配 summary 缓存版本或相关 endpoint 调整，支撑样本表新增 staleness 列。

关键符号：assemble_log_prob_from_cp, get_logits_and_tokens_offset_with_cp, slice_log_prob_with_cp, _cp_layout, _recover_cp_layout_legacy, _cp_widths_match, _build_train_row, _resolve_per_token_fields, save_debug_train_data, save_debug_train_data_for_rank, _PartitionReader._block, drawChart, renderTokens, loadTokensPane

关键源码片段

miles/dashboard/dump_reader.py

核心读取逻辑：在 load_joined 中按 shard 收集全部 handles 与位置索引，并通过 _cp_layout / _recover_cp_layout_legacy / _resolve_per_token_fields 按 PP/CP/TP 规则重组每个 per-token 字段到完整 response 长度，无法对齐的样本标记 alignment_failed。

```
# 只有这些 per-token 字段按 CP 切片存放；其余字段都是全长度
_CP_SHARDED_FIELDS = (
    "log_probs",
    "rollout_log_probs",
    "ref_log_probs",
    "entropy",
    "ref_entropy",
    "advantages",
    "returns",
)


def _resolve_per_token_fields(row: TrainRow, handles: list[dict], locations: list[tuple[int, int]]) ->
TrainRow:
    """把 `row` 的每个 per-token 字段填成完整 response 长度。

    统一在这里决定某个字段由哪个 shard 负责，因此 pipeline 并行（谁算的）、
    张量并行（冗余副本）和上下文并行（谁持有哪段切片）只处理一次，而不是
    每个视图各自实现一套可能出错的规则。
    """
    cp_size, layout, ok = _cp_layout(handles, locations, row.total_length, row.response_length)
    row.alignment_failed = not ok

    # 取首个 shard 作为主 shard，qkv_format 和 max_seq_len 用于 bshd 分块
```

```
    primary = handles[locations[0][0]]["rollout_data"]
    qkv_format = handles[locations[0][0]].get("qkv_format", "thd")
    max_seq_lens = primary.get("max_seq_lens")
    max_seq_len = None if max_seq_lens is None else int(max_seq_lens[locations[0][1]])

    for field in _CP_SHARDED_FIELDS:
        holders = [
            (shard, row_no) for shard, row_no in locations
            if handles[shard]["rollout_data"].get(field) is not None
        ]
        if not holders:
            # pipeline 并行下只有最后一个 stage 计算它，这里视为未 dump
            setattr(row, field, None)
            continue

        chunks: dict[int, torch.Tensor] = {}
        for shard, row_no in holders:
            if shard in layout:
                # setdefault：TP 副本共享同一个 cp_rank，保持选择顺序无关
                chunks.setdefault(layout[shard], handles[shard]["rollout_data"][field][row_no])

        if cp_size == 1:
            setattr(row, field, next(iter(chunks.values()), None))
            continue
        if len(chunks) != cp_size:
            # 切片不齐说明 dump 不完整，宁可置空并标记，也不显示错位数据
            setattr(row, field, None)
            row.alignment_failed = True
            continue

        setattr(
            row,
            field,
            assemble_log_prob_from_cp(
                chunks, row.total_length, row.response_length, cp_size,
                qkv_format=qkv_format, max_seq_len=max_seq_len,
            ),
        )

    # 矫正：任何字段长度不等于 response_length 都会与全长度 mask 错配，
    # 因此统一置空并标记 alignment_failed，而不是展示被截断 / 错位的值
    for field in _CP_SHARDED_FIELDS:
        values = getattr(row, field)
        if values is not None and len(values) != row.response_length:
            setattr(row, field, None)
            row.alignment_failed = True
    return row
```

miles/dashboard/store.py

修复 PartitonedReader._block 的读 - 改 - 写 race，加锁使 offset 读取、文件读取、解析和追加成为原子事务，消除 --follow 双计数。

```
  def _block(self, key: str, path: Path) -> Any:
      # 读文件 + 解析都会释放 GIL；不加锁时两个线程可能从同一个陈旧 offset
      # 读到相同字节并各自追加一次，导致重复记录。这里把 offset 读取、
      # 文件读取、解析和缓存更新包成一个事务。
      with self._lock:
          offset = self._offsets.get(key, 0)
          if key in self._blocks and path.stat().st_size <= offset:
              self._blocks.move_to_end(key)
              return self._blocks[key]

          with open(path, "rb") as f:
              f.seek(offset)
              chunk = f.read()
              end = chunk.rfind(b"\n")
              if end >= 0:
                  new = self._parse(chunk[: end + 1], path, offset)
                  pieces = [self._blocks[key], new] if key in self._blocks else [new]
                  self._blocks[key] = self._concat([piece for piece in pieces if self._length(piece)])
                  self._offsets[key] = offset + end + 1
              elif key not in self._blocks:
                  self._blocks[key] = self._concat([])
                  self._offsets[key] = offset

          self._blocks.move_to_end(key)
          while len(self._blocks) > self._max_blocks:
              evicted, _ = self._blocks.popitem(last=False)
              self._offsets.pop(evicted, None)
          return self._blocks[key]
```

评论区精华

PR 没有收到实质性的逐行 review 评论。Zhichenzzz 在最后提交后直接批准（LGTM），claude[bot] 仅自动提醒该仓库配置了手动 review。最有价值的反馈实际上来自提交历史：最后一个提交由 Zhichenzzz 完成，移除了 CP 组装测试中默认形式的 register_cpu_ci(...) 调用，因为 tests/ci/test/test_ci_register.py 断言禁止这种默认形式，该失败会卡住 stage-a-cpu 并导致整个 CI 报告 10 个 skipped lane，说明测试策略守卫会抓住隐藏的冗余配置。

- 整体批准与 CI 守卫修复 (other): PR 获批合并；CI 的 register_cpu_ci 调用被删除以保持测试 /CI 策略一致。
- CP 反向组装的设计对称性 (design): 采用共享偏移 helper 加 round-trip 测试，确保逆运算精确。

风险与影响

- 风险：

1. assemble_log_prob_from_cp 的逆运算精确性完全依赖 get_logits_and_tokens_offset_with_cp 的偏移语义不回退；如果未来修改 chunk 算法而未同步更新，round-trip 测试虽然会失败，但生产环境的错位告警可能更早出现。
2. 旧 dump 布局恢复采用 rollout_log_probs 内容匹配加宽度校验；若多个 rank 的切片恰好内容相同（如全零或常量），_recover_cp_layout_legacy 可能错误分组，_cp_widths_match 只能拦截宽度不匹配的误判。
3. _PartitionReader._block 加锁使所有分区读取串行化，--follow 高频轮询或大量并发请求时可能存在锁竞争，需要关注 dashboard 吞吐。
4. 前端图表改动（bands、gap）是纯视觉变更，没有自动视觉回归测试，回归风险依赖手动验证。
5. 新增 dump 字段改变了 train_data/*.pt 的 schema，旧 dump 通过 legacy 恢复，新 dump 若由旧版 reader 在部署回滚时读取会忽略新字段，行为仍安全。 - 影响：用户：使用 CP/PP 大规模并行训练（如 GB300 16 节点 GLM-5.2）时，dashboard 的 token 视图和 batch 视图首次可正常使用，不再出现错位或空列；--follow 模式计数恢复正确。系统：train dump 文件增加 3 个字段，磁盘占用可忽略；dashboard reader 对旧 dump 有兼容回退。团队：将 CP 切片与装配的偏移计算收敛到 cp_utils 单一来源，后续并行语义变更只需改一处；同时暴露了 CI 测试策略守卫对默认 register_cpu_ci 调用的严格约束。 - 风险标记：逆运算依赖偏移函数稳定性，旧 dump 布局推断有误判可能，分区读取加锁影响吞吐，前端图表改动缺视觉回归测试，新增 dump schema 需兼容旧数据

关联脉络

- PR #2589 Do not abort in-flight requests when bumping the engine weight version: 同一 rollout/weight-version 一致性链路，本 PR 的 staleness 统计正是围绕 weight_version 计算。
- PR #2544 Do not kill the run when one sample's collect_samples loses its connection: 同为 rollout/sample 数据链路上的容错与一致性修复，且都补充了针对断连 / 竞态的回归测试。