PR #2023 完整报告

radixark/miles

dashboard: dp-aware engine metrics

合并时间：2026-08-13 02:13

原文链接：http://prhub.com.cn/radixark/miles/pull/2023

执行摘要

- 一句话：引擎指标按 dp rank 正确折叠聚合，新增不均衡告警
- 推荐动作：值得精读。核心看点：(1) 指标聚合时机——折叠必须早于 rate/mean 派生，否则跨 rank diff 全是垃圾值；(2) 强度型 vs 累计型指标的差异化聚合（均值 vs 求和）；(3) 告警与运行配置联动的 _dp_spread_hint 设计，让告警直接指出当前 run 真正生效的开关；(4) 与 #2022 全窗口缓存合并时把缓存键升级为 (metric, per_dp_rank) 的并发正确性处理。tests/fast/dashboard/test_engine_dp.py 是这套聚合语义的极好文档，值得作为后续指标聚合改动的测试模板。

功能与动机

PR body 说明：dp-attention（如 --sglang-dp-size 4）下每个 dp rank 的 scheduler 会在 /metrics 导出各自的带标签序列，但 dashboard scraper 的 KEPT_LABELS 只保留 engine_type，结果 "every rank folded into one (addr, metric) series"，在真实 16 节点 GLM-5.2 run 上每个 (tick, engine) 恰好 4 个样本且 rank 身份丢失。由此造成：引擎图表 4 个 rank 值混在一条线（锯齿伪影、降采样后每点取任意 rank）；派生速率与延迟均值跨不同 rank 的计数器 diff，"mostly gaps and garbage"；wandb 快照 dashboard/engine_*_running_reqs 为 last-writer-wins；advisory 峰值并发最多低估 dp_size 倍；"the dp-rank starvation signal (one rank queueing while others idle — the actual congestion root cause on our run) was invisible"。

实现拆解

变更沿指标数据链路逐层展开，共 5 步：

1. 抓取层保留 rank 身份：miles/dashboard/sglang_scraper.py 的 KEPT_LABELS 从 ("engine_type",) 扩展为 ("engine_type", "dp_rank")，dp_rank 进入 EngineSample.labels。tests/fast/dashboard/test_scraper.py 新增 test_dp_rank_label_survives，断言 dp_rank 保留而 tp_rank、model_name 仍被过滤，白名单语义不变。
2. 存储层折叠与缓存（核心）：miles/dashboard/store.py 的 engine_series 增加 per_dp_rank=False 参数并透传到 _engine_parts / _engine_parts_cached。新增类常量 ENGINE_MEAN_AGGREGATED_METRICS（sglang_token_usage、sglang_cache_hit_rate、sglang_kv_transfer_latency_ms）区分强度型指标：折叠时取均值，其余（队列深度、吞吐、累计计数器）取求和。新增 _fold_dp_ranks：先把 labels_json 中的 dp_rank 与 null（chunk 级 struct 对齐产生）剥离，再按 (ts, addr,

metric, labels_json) 分组聚合。一次抓取会把一个 engine 内所有 rank 的样本打上同一个 scrape ts，因此按 ts 分组是精确的，同时顺带修复了保留 dp_rank 之前写入的旧 dump（同一 tick 存在多条 labels 完全相同的重复样本）。缓存键从 metric 升级为 (metric, per_dp_rank)，防止 per_dp_rank=True 调用拿到已折叠分区——这是与 main 分支 #2022 全窗口缓存冲突合并时的关键决策，折叠逻辑在缓存路径同样执行。

3. API 与采集对齐：miles/dashboard/server.py 的 timeline_engine_series 增加 per_dp_rank: bool = False 查询参数，默认关闭，既有消费者（metrics 视图、timeline overlay）响应结构不变。miles/dashboard/collector.py 的 latest_running_reqs 从 dict[str, float] 改为 dict[str, tuple[float, float]]（scrape ts, 同 tick 求和），_update_latest 在同 ts 时累加、跨 ts 时重置，wandb 快照 dashboard/engine*_running_reqs 不再 last-writer-wins。
4. 告警与前端：miles/dashboard/advisory.py 新增 DP_IMBALANCE_MIN_RUNNING = 1.0、DP_IMBALANCE_RATIO = 0.25，在 compute_advisories 末尾按 engine 统计各 rank 均值（_dp_rank_means），最忙 rank 有真实负载而最闲 rank 低于其 25% 时输出 warning；_dp_spread_hint 依据 run 的 args 快照指出真正生效的开关。miles/dashboard/args.py 将 use_miles_router、router_dp_aware、router_policy、sglang_router_policy、router_assignment_mode、sglang_load_balance_method 纳入 advisory 参数快照白名单。views_metrics.js 新增 "split by dp rank" 切换（sessionStorage 持久化），seriesLabel 使拆分模式标签变为 addr dpN，并让 legend 在数据到达前先渲染；views_timeline.js 顺带修复 legend 显示当前叠加指标名。
5. 测试与回放验证：新增 tests/fast/dashboard/test_engine_dp.py（6 例：gauge 跨 rank 求和、强度型 gauge 取均值、per_dp_rank=True 拆分、旧 dump 同 ts 重复折叠、counter 先折叠再差分得 80/s、histogram 均值按 count 加权得 11/5 而非 mean-of-means）；test_advisory.py 新增 4 例（触发、四种开关文案、均衡不报、空闲不报）。PR body 报告 193 个 dashboard 测试全数通过，并对真实 dp4 run 的 dump 回放：4× 重复样本折叠为每条 engine 一条序列，gen-token 速率恢复 27.6k 点。

关键文件：

- miles/dashboard/store.py（模块 指标存储；类别 source；类型 core-logic；符号 engine_series, _engine_parts, _fold_dp_ranks, _engine_parts_cached）：核心语义所在：engine_series 新增 per_dp_rank 参数，新增 _fold_dp_ranks 按 scrape ts 折叠各 rank 采样（累计型求和、强度型均值），并修复旧 dump 同 ts 重复；分区缓存键升级为 (metric, per_dp_rank)。
- tests/fast/dashboard/test_engine_dp.py（模块 指标聚合；类别 test；类型 test-coverage；符号 make_store, dp_samples, test_gauge_sums_across_dp_ranks, test_intensive_gauge_averages_across_dp_ranks）：新增测试文件，6 个用例精确固定聚合语义：求和、均值、拆分、旧数据修复、先折叠再差分的速率、按 count 加权的直方图均值，是本 PR 语义的最佳文档。
- miles/dashboard/advisory.py（模块 告警分析；类别 source；类型 core-logic；符号 _dp_rank_means, _dp_spread_hint, compute_advisories, DP_IMBALANCE_MIN_RUNNING）：新增 dp 不均衡告警：_dp_rank_means 按 engine 求各 rank 均值，_dp_spread_hint 依据 run 的 args 快照指出实际生效的路由开关；直接

回应评审关于负载均衡策略的追问。

- tests/fast/dashboard/test_advisory.py（模块 告警测试；类别 test；类型 test-coverage；符号 _dp_engine, _imbalanced, test_dp_imbalance_names_the_knob_that_applies, test_dp_balanced_not_flagged）：新增 4 个 dp 告警用例，覆盖触发条件、四种开关文案、均衡不报、空闲不报，验证告警与 args 快照联动。
- miles/dashboard/static/views_metrics.js（模块 前端图表；类别 source；类型 core-logic；符号 seriesLabel, perDpRank, renderLegend, refresh）：前端新增 split by dp rank 切换（sessionStorage 持久化）与 seriesLabel，拆分模式下标签变为 addr dpN，legend 提前渲染避免切换后空白。
- miles/dashboard/server.py（模块 接口服务；类别 source；类型 core-logic；符号 timeline_engine_series）：timeline_engine_series 透传 per_dp_rank 参数，默认 False 保持既有消费者响应结构不变。
- miles/dashboard/collector.py（模块 指标采集；类别 source；类型 core-logic；符号 _update_latest, _latest_running_reqs）：_latest_running_reqs 从单值改为 (ts, sum)，同一次 scrape 的多个 rank 样本求和，wandb 快照不再 last-writer-wins；Zhichenzzz 的注释精简意见也在此文件。
- tests/fast/dashboard/test_scraper.py（模块 抓取测试；类别 test；类型 test-coverage；符号 test_dp_rank_label_survives）：新增 test_dp_rank_label_survives，验证 dp_rank 标签经 scraper 保留而 tp_rank/model_name 仍被过滤。
- miles/dashboard/sglang_scraper.py（模块 指标抓取；类别 source；类型 core-logic；符号 KEPT_LABELS）：KEPT_LABELS 加入 dp_rank，rank 身份进入 EngineSample.labels，是整条链路的修复起点。
- miles/dashboard/args.py（模块 参数配置；类别 source；类型 configuration）：advisory 参数快照白名单新增 router 相关键，供 _dp_spread_hint 依据真实 run 配置给出准确开关。
- miles/dashboard/static/views_timeline.js（模块 前端图表；类别 source；类型 core-logic）：顺带修复：timeline legend 显示当前叠加的 overlay 指标名，随本 PR 一并合入。

关键符号：engine_series, _fold_dp_ranks, _engine_parts, _engine_parts_cached, _dp_rank_means, _dp_spread_hint, compute_advisories, timeline_engine_series, _update_latest, seriesLabel

关键源码片段

miles/dashboard/store.py

核心语义所在：engine_series 新增 per_dp_rank 参数，新增 _fold_dp_ranks 按 scrape ts 折叠各 rank 采样（累计型求和、强度型均值），并修复旧 dump 同 ts 重复；分区缓存键升级为 (metric, per_dp_rank)。

```
# miles/dashboard/store.py —— dp rank 折叠与缓存（核心实现）

class MetricStore:
    # 强度型指标跨 dp rank 折叠时取均值，其余（队列深度、吞吐、累计计数器）取求和
    ENGINE_MEAN_AGGREGATED_METRICS: ClassVar[frozenset[str]] = frozenset(
```

```
    {"sglang_token_usage", "sglang_cache_hit_rate", "sglang_kv_transfer_latency_ms"}
)

def engine_series(
    self,
    metric: str,
    *,
    t0: float | None = None,
    t1: float | None = None,
    max_points: int = 2000,
    per_dp_rank: bool = False,
) -> list[dict]:
    """每个 (addr, label set) 一条序列；默认先折叠再派生，per_dp_rank=True 保留 rank
    维度。"""
    if metric in self.ENGINE_RATE_METRICS:
        parts = self._engine_parts(self.ENGINE_RATE_METRICS[metric], t0, t1, per_dp_rank=
        per_dp_rank)
        return self._derived_series(parts, None, max_points)
    if metric in self.ENGINE_MEAN_METRICS:
        base = self.ENGINE_MEAN_METRICS[metric]
        return self._derived_series(
            self._engine_parts(base + "_sum", t0, t1, per_dp_rank=per_dp_rank),
            self._engine_parts(base + "_count", t0, t1, per_dp_rank=per_dp_rank),
            max_points,
        )
    out = []
    for (addr, labels_json), part in self._engine_parts(metric, t0, t1, per_dp_rank=per_dp_rank)
    .items():
        ts, values = stride_downsample(part["ts"].to_numpy(), part["value"].to_numpy(), max_
        points)
        out.append(dict(addr=addr, labels=_engine_labels(labels_json), ts=ts.tolist(), value=
        values.tolist()))
    return out

def _fold_dp_ranks(self, frame: pl.DataFrame, metric: str) -> pl.DataFrame:
    """Engine-level 视图：把各 dp rank 采样折叠成 engine 级序列。

    一次抓取会同时拿到一个 engine 内所有 rank 的样本（scrape ts 相同），
    因此按 ts 分组精确；同时修复保留 dp_rank 之前写入的旧 dump——同一 tick
    存在多条 labels 完全相同的重复样本。
    """
    if frame.is_empty():
        return frame
    # 剥离 dp_rank 并清理 null：chunk 级 struct 对齐会用 null 补齐缺失标签，
    # 不清理会把一条逻辑序列拆成多条
    stripped = {
        labels_json: json.dumps(
            {k: v for k, v in json.loads(labels_json).items() if k != "dp_rank" and v is not None},
            sort_keys=True,
```

```
            )
            for labels_json in frame["labels_json"].unique()
        }
        frame = frame.with_columns(pl.col("labels_json").replace_strict(stripped))
        agg = pl.col("value").mean() if metric in self.ENGINE_MEAN_AGGREGATED_METRICS else
        pl.col("value").sum()
        return frame.group_by(["ts", "addr", "metric", "labels_json"]).agg(agg.alias("value"))

    def _engine_parts_cached(self, metric: str, per_dp_rank: bool) -> dict[tuple[str, str], pl.
    DataFrame]:
        # 缓存键必须是 (metric, per_dp_rank)：否则 per_dp_rank=True 的调用可能
        # 拿到已折叠分区，反之亦然（与 #2022 全窗口缓存合并时确认的关键点）
        frame, cache = self._engine_cache()
        cache_key = (metric, per_dp_rank)
        parts = cache.get(cache_key)
        if parts is None:
            frame = frame.filter(pl.col("metric") == metric)
            if not per_dp_rank:
                frame = self._fold_dp_ranks(frame, metric)
            parts = {
                key: part.sort("ts")
                for key, part in sorted(frame.partition_by(["addr", "labels_json"], as_dict=True).
                items())
            }
            cache[cache_key] = parts
        return parts
```

tests/fast/dashboard/test_engine_dp.py

新增测试文件，6 个用例精确固定聚合语义：求和、均值、拆分、旧数据修复、先折叠再差分的速率、按 count 加权的直方图均值，是本 PR 语义的最佳文档。

```
# tests/fast/dashboard/test_engine_dp.py —— 用测试固定聚合语义（先折叠再派生的直观例证）

def dp_samples(metric, points_by_rank):
    """按 rank 生成采样，rank 身份写入 labels，模拟 dp-attention 抓取结果。"""
    return [
        EngineSample(ts=ts, addr=ADDR, metric=metric, labels={"dp_rank": rank}, value=v)
        for rank, points in points_by_rank.items()
        for ts, v in points
    ]


def test_counter_rate_sums_ranks_before_diff(tmp_path):
    # 两个 rank 的生成 token 计数器：2 秒内各自 0 -> 100 与 0 -> 60。
    # engine 级速率必须是总和 80/s，而不是跨 rank 交错 diff 出的垃圾值
    store = make_store(
        tmp_path,
        dp_samples(
            "sglang_generation_tokens_total",
```

```
            {"0": [(0.0, 0.0), (2.0, 100.0)], "1": [(0.0, 0.0), (2.0, 60.0)]},
        ),
    )
    (series,) = store.engine_series("sglang_generation_tokens_per_s")
    assert series["ts"] == [2.0]
    assert series["value"] == [80.0]
    # 拆分模式下每个 rank 独立成线，便于定位具体 rank
    per_rank = store.engine_series("sglang_generation_tokens_per_s", per_dp_rank=True)
    assert {s["labels"]["dp_rank"]: s["value"] for s in per_rank} == {"0": [50.0], "1": [30.0]}


def test_histogram_mean_weights_ranks_by_count(tmp_path):
    # rank 0：4 次完成共 8 s；rank 1：1 次完成 3 s。先折叠 sum 与 count
    # 再求 delta 比，得到 11 s / 5 次 = 2.2 s，而不是 mean-of-means
    samples = dp_samples(
        "sglang_time_to_first_token_seconds_sum",
        {"0": [(0.0, 0.0), (2.0, 8.0)], "1": [(0.0, 0.0), (2.0, 3.0)]},
    ) + dp_samples(
        "sglang_time_to_first_token_seconds_count",
        {"0": [(0.0, 0.0), (2.0, 4.0)], "1": [(0.0, 0.0), (2.0, 1.0)]},
    )
    store = make_store(tmp_path, samples)
    (series,) = store.engine_series("sglang_ttft_mean_s")
    assert series["ts"] == [2.0]
    assert series["value"] == [11.0 / 5.0]
```

miles/dashboard/advisory.py

新增 dp 不均衡告警：_dp_rank_means 按 engine 求各 rank 均值，_dp_spread_hint 依据 run 的 args 快照指出实际生效的路由开关；直接回应评审关于负载均衡策略的追问。

```
# miles/dashboard/advisory.py —— dp 不均衡告警：阈值、按 engine 求各 rank 均值、
# 以及按 run 配置指出真正生效的路由开关
DP_IMBALANCE_MIN_RUNNING = 1.0 # 最忙 rank 必须承载的真实负载下限
DP_IMBALANCE_RATIO = 0.25 # 最闲 rank 低于最忙 rank 该比例即视为失衡


def _dp_rank_means(series: list[dict]) -> dict[str, dict[str, float]]:
    """从 per_dp_rank=True 的结果中提取每个 engine 的 {dp_rank: 均值}；
    没有 dp_rank 标签的序列（非 dp engine）直接跳过。"""
    out: dict[str, dict[str, float]] = {}
    for s in series:
        rank = s["labels"].get("dp_rank")
        if rank is None or not s["value"]:
            continue
        out.setdefault(s["addr"], {})[rank] = sum(s["value"]) / len(s["value"])
    return out


def _dp_spread_hint(args: dict) -> str:
```

```
    """指出当前 run 中真正决定请求如何跨 dp rank 分布的开关。

    --router-dp-aware 只有在 sglang router 本身已 dp-aware 时才生效;
    --use-miles-router 下该 flag 失效，转由 sglang 的 dp controller 派发。
    """
    if args.get("use_miles_router"):
        return "the miles router routes per engine; --sglang-load-balance-method decides the
        rank"
    if not args.get("router_dp_aware"):
        return "the router sees one worker per engine; --router-dp-aware makes it route per rank"
    policy = args.get("sglang_router_policy") or args.get("router_policy")
    if policy == "manual":
        return (
            f"manual routing pins each key to a rank via --router-assignment-mode "
            f"({args.get('router_assignment_mode')}); min_load spreads by load"
        )
    return f"--router-policy ({policy}) picks the rank"


# compute_advisories 末尾的失衡检查（节选）：
# 窗口内按 engine 取各 rank 运行中请求的均值，命中阈值才给出带具体开关的告警
per_rank = store.engine_series("sglang_num_running_reqs", t0=t0, t1=t1, per_dp_rank=True)
for addr, means in sorted(_dp_rank_means(per_rank).items()):
    if len(means) < 2:
        continue
    busiest, idlest = max(means.values()), min(means.values())
    if busiest >= DP_IMBALANCE_MIN_RUNNING and idlest < DP_IMBALANCE_RATIO * busiest:
        detail = ", ".join(
            f"dp{rank}={mean:.1f}" for rank, mean in sorted(means.items(), key=lambda kv: int(kv[0]
            ))
        )
        out.append(
            Advisory(
                level="warning",
                message=(
                    f"{addr}: dp ranks imbalanced (mean running reqs {detail}) — requests pile onto "
                    f"few ranks while others idle; {_dp_spread_hint(args)}"
                ),
            )
        )
```

评论区精华

两轮 review 评论均已解决，PR 最终 APPROVED：

1. advisory.py：Zhichenzzz 追问 "also checking load balance strategy (maybe manual + min_load?)"——初版告警只提 --router-dp-aware 不够全面。作者用 commit a413488 将告警升级为 _dp_spread_hint：按 run 的 args 快照分四种情形给出实际生效开关（--use-miles-router → --sglang-load-balance-method；未开 router_dp_aware →

--router-dp-aware；manual → --router-assignment-mode；否则 --router-policy），并新增 test_dp_imbalance_names_the_knob_that_applies 覆盖四个分支。

2. collector.py：Zhichenzzz 提 "trim those comments?"，作者以 commit 4878419 "drop the running-reqs cache comment" 删除冗余注释，行为不变。

- dp 不均衡告警应指向实际生效的路由开关 (design): 作者新增 _dp_spread_hint 按 args 快照分四种情形给出正确开关（--use-miles-router → --sglang-load-balance-method；未开 router_dp_aware → --router-dp-aware；manual → --router-assignment-mode；否则 --router-policy），并以 test_dp_imbalance_names_the_knob_that_applies 覆盖四个分支。
- collector 缓存注释精简 (style): commit 4878419 "drop the running-reqs cache comment" 删除冗余注释，行为不变。

风险与影响

- 风险：
  1. 折叠依赖同 ts 假设：_fold_dp_ranks 按浮点 ts 精确分组，成立前提是一次抓取内所有 rank 样本共享同一 scrape ts。若未来抓取改为并发或 engine 端时间戳出现抖动，折叠会静默退化为拆分。当前实现成立，但建议后续在写入路径增加同 tick 时间戳归一化作为防御。
  2. 旧 dump 修复语义：无 dp_rank 标签的同 ts 重复样本从 last-writer-wins 改为求和。对 running_reqs 这类 gauge 语义是修正；但若历史 dump 中同 ts 重复源于重试或重复写入而非 dp rank，求和会放大数值。仅影响旧数据回放。
  3. 派生指标数值量级变化：engine 级速率从跨 rank 交错 diff 的垃圾值变为各 rank 速率之和，histogram 均值变为按 count 加权，数值会增大到真实总量，属预期修正；任何依赖旧错误数值的自动化断言可能被打破。
  4. 启发式阈值误报：DP_IMBALANCE_MIN_RUNNING = 1.0 与 DP_IMBALANCE_RATIO = 0.25 未参数化，短时突发可能误报；已有 balanced/idle 用例抑制低负载噪声，但阈值仍需真实 run 数据校准。
  5. 测试覆盖缺口：collector.py 的同 tick 求和逻辑没有新增直接单测（本 PR 的 scraper 测试只覆盖标签保留）；缓存键升级依赖现有 _engine_cache_lock 并发语义，折叠 group_by 在锁内缓存构建期执行，量级可控。
- 影响：
  1. 用户（dashboard 使用者）：dp-attention 部署的引擎图表从锯齿混线变为真实聚合曲线；gen-token 速率等派生指标恢复完整覆盖（真实 dump 回放 27.6k 点）；wandb dashboard/engine_*_running_reqs 快照从任意一个 rank 变为同 tick 求和；advisory 能直接指出 rank 饥饿，并按当前 run 的 router 配置给出正确的排障开关。
  2. 系统：store 每次折叠增加一次 group_by，带缓存路径（无窗口查询）只付一次成本；API 保持向后兼容（per_dp_rank 默认 False，响应结构不变）；前端 split by dp rank 状态存 sessionStorage，刷新保留。
  3. 团队：为后续 PD prefill/decode、多实例指标的跨实例聚合提供了可复用范式（先折叠再派生、强度型 / 累计型差异化聚合）；advisory 首次把告警文案与 run 的实际 args 快照联动，该思路可推广到其他告警。 - 风险标记：核心存储路径变更，折叠聚合语义变

更，旧数据修复依赖同 ts 假设，启发式告警阈值误报，collector 求和缺直接单测

关联脉络

- PR #2353 dashboard: report model FLOPs utilization: 同改 miles/dashboard/advisory.py，merge commit 明确记录与 #2353 的 MFU 告警规则冲突并做纯增量合并；两条告警线（MFU 与 dp 不均衡）共享 compute_advisories 框架。
- PR #2027 dashboard: advisory v2 — run-health alarms before config tuning: 同属 advisory 演进线：先确立 run-health 告警优先与门控，本 PR 再叠加 dp 不均衡告警，test_advisory.py 持续扩充。
- PR #2026 dashboard: scrape engines directly by default: 同改 scraper/collector 管线（sglang_scraper.py、collector.py、test_scraper.py 等），本 PR 延续抓取语义修正：保留 dp_rank 标签与同 tick 求和。
- PR #2024 dashboard: read open phase markers regardless of age: 同改 miles/dashboard/store.py 的引擎数据读取路径，本 PR 的 _engine_parts/_engine_parts_cached 与相位读取修复属于 store 层同区域演进。