PR #2022 完整报告

radixark/miles

dashboard: cache full-window engine series queries

合并时间：2026-08-13 01:52

原文链接：http://prhub.com.cn/radixark/miles/pull/2022

执行摘要

- 一句话：缓存 engine_series 全窗口查询，指标页从秒级降到毫秒级
- 推荐动作：值得精读。核心看点在 _engine_cache() 的 " 版本最后赋值 + 成对返回 frame/partitions + 单锁 " 设计：如何在 FastAPI sync 端点多线程并发读路径上安全地做惰性缓存重建，是可直接复用的并发模式。另一个值得关注的设计决策是缓存只覆盖全窗口路径、有界窗口保留懒加载，避免了为短窗口查询引入缓存一致性复杂度。建议阅读顺序：先看 review 评论中的竞态分析，再对照 commit 2（1e3e66ea）如何落地修复，最后看测试如何用并发线程和失败注入覆盖这两个正确性保障。

功能与动机

PR body 明确指出打开 sglang metrics 标签页耗时数秒：页面按 metric key 发约 15 个 /api/timeline/engine_series 请求，每个请求都独立重新拼接每个小时 partition 块并重新过滤整个 engine_series 流（9.2M 行 / 1.4GB），衍生指标（ttft/e2e/rate 族）还要做两次（_sum + _count）。0.3–4s 每请求排队在浏览器每来源 6 连接限制之后，标签页需要数秒才能加载，并且每次 poll tick 都会重新付出全部成本。

实现拆解

1. 缓存状态注册：在 miles/dashboard/store.py 的 MetricStore.__init__中新增 threading.Lock、版本号元组、全窗口 frame 缓存与 per-metric partition 缓存字典四个字段，并 import threading。
2. 全窗口缓存重建入口：新增 _engine_cache()，以 reader.files() 遍历所有 partition 文件得到的 (key, st_size) 元组作为版本号；在锁内比对版本，不一致时调用 reader.window(None, None) 重建全窗口 frame 并清空 partition 缓存字典，版本号最后赋值——重建失败时版本不更新，失败状态不会被误认为已缓存。
3. per-metric 分区惰性构建：新增 _engine_parts_cached(metric)，从缓存 frame 中 filter 出对应 metric 并按 (addr, labels_json) partition_by 后排序，构建结果按 metric 名惰性写入缓存字典，避免一次构建全部 metric。
4. 调用点接入：_engine_parts() 在 t0 与 t1 均为 None 时（metrics 页面请求的形态）改走 _engine_parts_cached()；engine_metric_names() 原先每次全流扫描，现复用 _engine_cache() 返回的 frame。有界窗口（timeline overlay）查询保留原有懒加载 per-partition 路径，不动其语义。

5. 测试配套：tests/fast/dashboard/test_engine_derived.py 新增两个回归用例——test_full_window_cache_survives_concurrent_cold_readers 用 8 线程并发冷读验证只重建一次且结果一致；test_failed_rebuild_does_not_look_cached 用 monkeypatch 让 reader.window 抛 OSError，验证失败重建不污染缓存、恢复后能正常读取。PR body 提到的 14 passed 对应 dashboard 测试集，实际新增用例落在 test_engine_derived.py。

关键文件：

- miles/dashboard/store.py（模块 指标存储；类别 source；类型 core-logic；符号 _engine_cache, _engine_parts_cached, _engine_parts, engine_metric_names）：核心改动文件：MetricStore 新增全窗口缓存状态、锁、版本化失效逻辑，接入 engine_series 与 engine_metric_names 两个读路径，是本次性能提升与并发正确性的全部载体。
- tests/fast/dashboard/test_engine_derived.py（模块 缓存测试；类别 test；类型 test-coverage；符号 test_full_window_cache_survives_concurrent_cold_readers, test_failed_rebuild_does_not_look_cached, _collect, _raise）：新增两个针对性回归测试：并发冷读一致性测试验证锁与单次重建语义，失败重建测试验证版本最后赋值不被污染，是并发正确性设计的关键保障。

关键符号：_engine_cache, _engine_parts_cached, _engine_parts, engine_metric_names, engine_series

关键源码片段

miles/dashboard/store.py

核心改动文件：MetricStore 新增全窗口缓存状态、锁、版本化失效逻辑，接入 engine_series 与 engine_metric_names 两个读路径，是本次性能提升与并发正确性的全部载体。

```
def _engine_cache(self) -> tuple[pl.DataFrame, dict[str, dict[tuple[str, str], pl.DataFrame]]]:
    """全窗口 frame 与其分区缓存，成对返回：并发调用者基于同一 frame 写
    partition，避免跨版本错配。"""
    reader = self._readers[Stream.ENGINE_SERIES]
    with self._engine_cache_lock:
        # 版本号 = 每个 partition 文件的 (key, st_size) 元组；append-only
        追加会改变文件大小，从而自动失效
        version = tuple((key, path.stat().st_size) for key, path in reader.files())
        if version != self._engine_cache_version:
            self._engine_frame_cache = reader.window(None, None)
            self._engine_parts_cache = {}
            # 版本最后赋值：重建失败时不更新版本，失败状态不会被视为已缓存
            self._engine_cache_version = version
        return self._engine_frame_cache, self._engine_parts_cache


def _engine_parts_cached(self, metric: str) -> dict[tuple[str, str], pl.DataFrame]:
    """按 metric 惰性构建 partitions，命中后直接返回排序好的 part 字典。"""
    frame, cache = self._engine_cache()
    parts = cache.get(metric)
```

```
    if parts is None:
        frame = frame.filter(pl.col('metric') == metric)
        parts = {
            key: part.sort('ts')
            for key, part in sorted(frame.partition_by(['addr', 'labels_json'], as_dict=True).items())
        }
        cache[metric] = parts
    return parts


def _engine_parts(self, metric: str, t0: float | None, t1: float | None) -> dict[tuple[str, str], pl.
DataFrame]:
    # 只有全窗口查询走缓存 —— 这正是 metrics 页面请求的形态；有界 timeline 查询保留原懒加载
    per-partition 路径
    if t0 is None and t1 is None:
        return self._engine_parts_cached(metric)
    frame = self._window(self._readers[Stream.ENGINE_SERIES].window(t0, t1), t0, t1).filter(
        pl.col('metric') == metric
    )
    return {
        key: part.sort('ts')
        for key, part in sorted(frame.partition_by(['addr', 'labels_json'], as_dict=True).items())
    }
```

tests/fast/dashboard/test_engine_derived.py

新增两个针对性回归测试：并发冷读一致性测试验证锁与单次重建语义，失败重建测试验证版本最后赋值不被污染，是并发正确性设计的关键保障。

```
def test_full_window_cache_survives_concurrent_cold_readers(tmp_path):
    """8 个线程并发冷读：模拟 FastAPI threadpool 下多请求同时触发首次缓存重建。"""
    store = make_store(tmp_path, counter('http://e:1', [(0.0, 0.0), (2.0, 100.0)]))
    reader = store._readers[Stream.ENGINE_SERIES]
    real_window = reader.window
    # 注入 0.2s 慢读取，放大重建窗口，让其他线程更容易撞上未就绪状态
    reader.window = lambda t0, t1: (time.sleep(0.2), real_window(t0, t1))[1]

    results, errors = [], []
    threads = [
        threading.Thread(target=lambda: _collect(store, results, errors), name=f'reader-{i}')
        for i in range(8)
    ]
    for thread in threads:
        thread.start()
    for thread in threads:
        thread.join()
    assert errors == []
    # 所有线程必须读到同一份结果：只有一次 0.2s 延迟重建，而不是各自独立重建
    assert all(r == results[0] and r[0]['value'] == [50.0] for r in results)
```

评论区精华

Zhichenzzz 在初版实现上发现了并发竞态并给出了统一修复结论：数据端点都是 sync def，FastAPI 在 anyio threadpool 中并发执行，正是 15 请求同时打开标签页的场景。初版先写版本号再重建 frame，线程 A 在 window(None, None) 内耗时 0.3–4s 时，线程 B..O 看到版本匹配便跳过重建，直接返回仍为 None 的 _engine_frame_cache，_engine_parts_cached 对 None 调 .filter() 抛 AttributeError，且该异常不在 _translate_errors() 映射内，会以裸 500 上抛。隐藏的第二个竞态是：从 frame V1 算出的 parts 可能写入另一线程刚安装的 V2 字典，stale partitions 会一直被服务到下次版本递增。一个锁同时解决两个问题并序列化 rebuild，版本赋值必须移到最后。提交 1e3e66ea 按此修复，PR 随后获 APPROVED。

- engine 缓存重建竞态：冷读拿到 None frame 与 stale partitions (correctness): 一个锁同时解决两个问题并序列化 rebuild；版本赋值必须移到最后，失败时不更新版本避免污染缓存。提交 1e3e66ea 按此修复，PR 随后获 APPROVED。

风险与影响

- 风险：
  1. 缓存失效依赖文件大小版本：版本号只取 partition 文件的 st_size，依赖 append-only 写入约定（追加必然改变大小）；若文件被非常规原地覆盖且字节数不变，缓存不会失效。跨进程下 collector 写入与 server 读取的 stat 可见性一般是即时的，但极端时序下可能多服务一次旧数据。
  2. 内存占用上升：全窗口 frame（9.2M 行量级）常驻内存，叠加按 metric 构建的 partition 视图（约 15 个 metric 的 dict 副本），大 dump + live/follow 长期运行场景 RSS 可能接近翻倍，需要观察。
  3. 全局锁串行化：_engine_cache_lock 是全局锁，首次冷启动 rebuild 的 0.3–4s 内，其余 engine 全窗口查询都会排队等待；这是换取一致性后的可接受代价，但应知悉。
  4. 错误翻译缺口：AttributeError 不在 server.py 的 _translate_errors() 映射中，当前竞态虽被锁消除，但缓存路径若再引入类型错误仍会以裸 500 呈现，错误映射表本身值得补全。
  5. 有界窗口无收益：timeline overlay 的短窗口查询不缓存，这是有意的设计取舍而不是回归，但若未来出现高频有界查询，需要另做按 partition 键粒度的缓存。 - 影响：对用户：sglang metrics 标签页的打开与轮询从数秒级降至毫秒级（15 并发 6.5s → 0.39s，warm 单请求 0.3–4s → 约 10ms），每次 poll tick 不再重扫 1.4GB 流。对系统：读路径的 CPU/IO 开销大幅下降，但带来常驻内存上升与首次冷启动的短暂锁等待。对团队：store.py 读路径新增了明确的并发正确性约束（版本顺序、锁、pair 返回），后续改动需保持这些语义；同时补充了两个针对并发回归的测试用例，可作为类似缓存设计的参考模板。 - 风险标记：缓存一致性依赖文件大小版本，全窗口 frame 常驻内存，全局锁串行化冷启动，错误翻译映射未覆盖 AttributeError

关联脉络

- PR #2026 dashboard: scrape engines directly by default: 同属 engine 指标数据管线演进，解决引擎指标静默缺失问题并默认直采，与本 PR 一起保证 sglang metrics 数据的可

观测性与可查询性能。

- PR #2023 dashboard: dp-aware engine metrics: 同样修改 store.py 并在 engine 指标语义上做聚合修正（按 dp rank 折叠），与本 PR 的 engine_series 缓存共享同一读路径与 MetricStore 结构。
- PR #2476 fix(dashboard): zero the trainer log-probs the loss masks out: dashboard 指标展示正确性修复，与本 PR 同为 dashboard 观测体验优化线，但作用在 dump_reader 侧而非 store 缓存侧。
- PR #2024 dashboard: read open phase markers regardless of age: 同样改动 miles/dashboard/store.py 读取层行为（移除相位读取下界），显示 store 读路径近期持续演进。