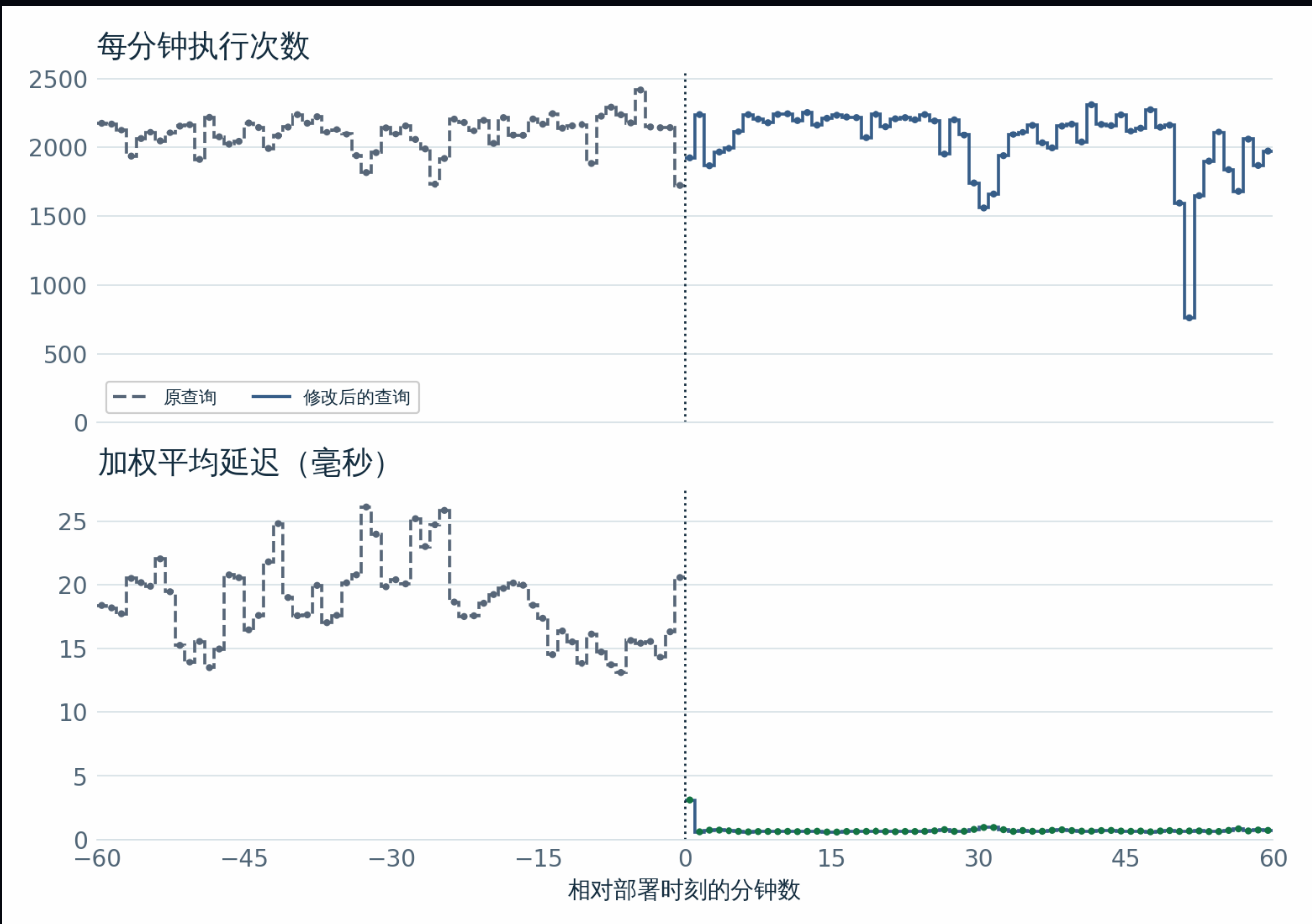
MariaDB 18.43 0.70

Aurélien LEQUOY · 2026 11 21

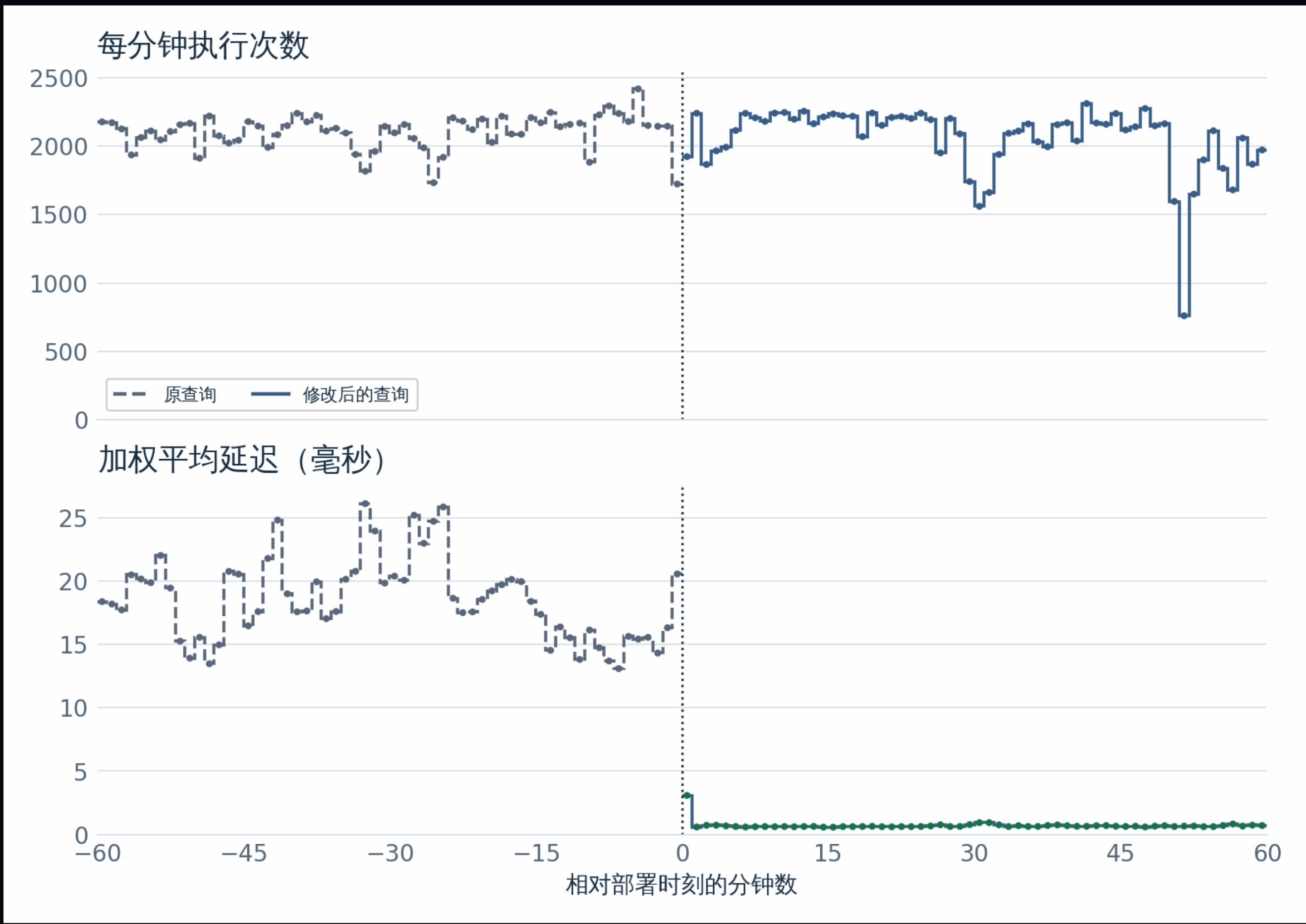
PMACONTROL MARIADB PERFORMANCE SQL BENCHMARK OBSERVABILITY



`ts_variable` **3,791** **3,759** `name` `mysql_available` `v.`from` = 'mysql_server'` **18.43** **0.70** MariaDB

PmaControl **2326** **SQL 87**

2026 11 21 17:26:47 SQL Performance Schema



每组数据有 120 个点，每分钟一个。上图显示每分钟的执行次数，下图显示同一批调用的平均耗时。竖向虚线表示部署时刻。切换前后少量来自另一版本的调用也计入统计。

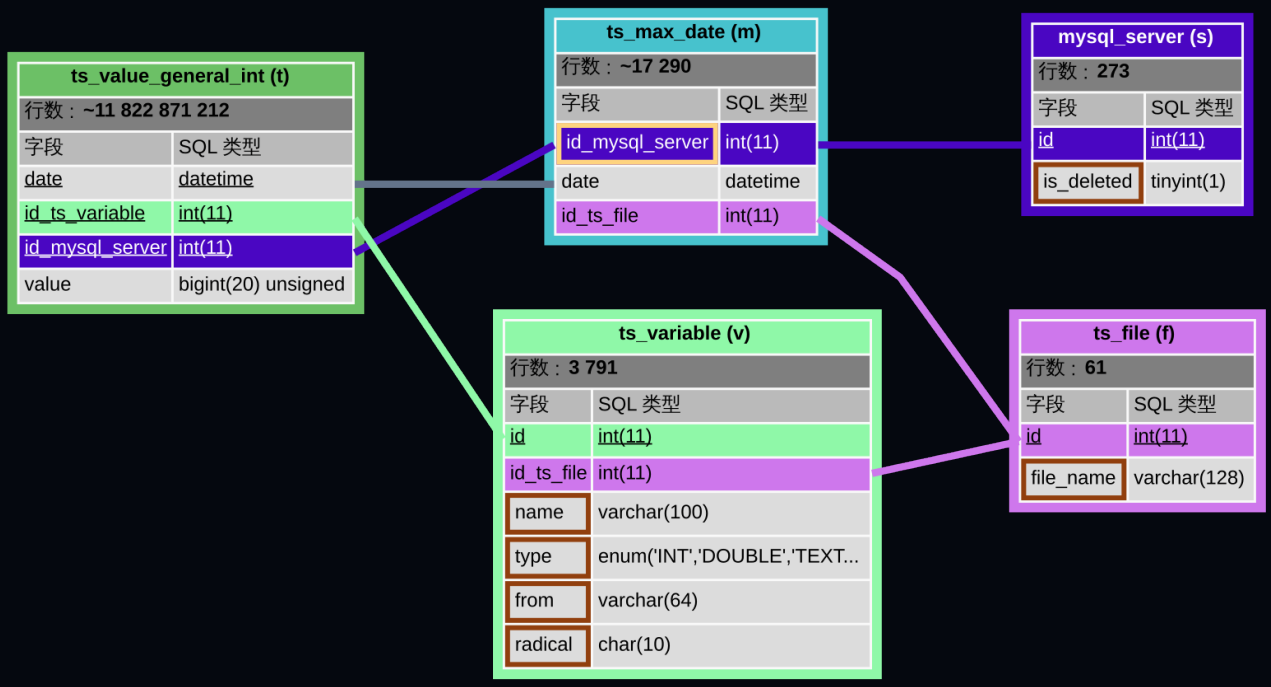
表格中，较大值严格超过较小值的两倍时，绿色（✓）表示改善，红色（✗）表示退化。对于耗时，绝对差还须至少为 1 毫秒。调用量用于说明负载。此阈值用于辅助阅读。图中以不同的线条区分两个时段。部署后每个延迟点都与前一小时

指标	部署前一小时	部署后一小时
估算执行次数	126,229	123,241
每分钟执行次数	2,104	2,054
加权平均延迟	18.43 ms	✓ 0.70 ms (÷26.25)
累计 SQL 秒数	2,326.5	✓ 86.5 (÷26.88)
每 SQL 秒执行次数	54	✓ 1,424 (×26.25)
每次调用检查行数	2.99	✓ 1.00 (÷2.98)

按每秒累计 SQL 次执行时间衡量，降低 14296.12%。调用量保持稳定，因此该比率主要反映单次读取的成本，实际吞吐量还取决于轮询频率和并发度。

这条查询读取什么

ts_max_date ts_file ts_variable mysql_server NULL



PmaControl ts_variable.from

MariaDB TABLE ts_variable UNIQUE KEY

SHOW CREATE TABLE ts_variable;

```
CREATE TABLE `ts_variable` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `id_ts_file` int(11) NOT NULL DEFAULT 3,
  `name` varchar(100) NOT NULL,
  `type` enum('INT','DOUBLE','TEXT','JSON','') NOT NULL COMMENT '1 => int, 2 => double, 3 =>
text, 4=> json',
  `from` varchar(64) NOT NULL,
  `radical` char(10) NOT NULL,
  `is_derived` int(11) DEFAULT 1,
  `is_dynamic` int(11) DEFAULT 1 COMMENT 'only for global variables',
  PRIMARY KEY (`id`),
  UNIQUE KEY `name` (`name`,`from`),
  KEY `ts_variable_ibfk_1` (`id_ts_file`),
  KEY `from` (`from`),
  CONSTRAINT `ts_variable_ibfk_1` FOREIGN KEY (`id_ts_file`) REFERENCES `ts_file` (`id`) ON
DELETE CASCADE ON UPDATE CASCADE
```

```
) ENGINE=InnoDB AUTO_INCREMENT=39387 DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;
```

SHOW CREATE TABLE ts_file;

```
CREATE TABLE `ts_file` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `file_name` varchar(128) NOT NULL,
  `each` int(11) NOT NULL DEFAULT 1,
  PRIMARY KEY (`id`),
  UNIQUE KEY `file_name` (`file_name`)
) ENGINE=InnoDB AUTO_INCREMENT=25508473 DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;
```

SHOW CREATE TABLE ts_max_date;

```
CREATE TABLE `ts_max_date` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `id_mysql_server` int(11) NOT NULL,
  `date` datetime DEFAULT NULL,
  `sample_epoch` bigint(20) unsigned NOT NULL DEFAULT 0,
  `date_p1` datetime DEFAULT NULL,
  `sample_epoch_p1` bigint(20) unsigned NOT NULL DEFAULT 0,
  `date_p2` datetime DEFAULT NULL,
  `sample_epoch_p2` bigint(20) unsigned NOT NULL DEFAULT 0,
  `date_p3` datetime DEFAULT NULL,
  `sample_epoch_p3` bigint(20) unsigned NOT NULL DEFAULT 0,
  `date_p4` datetime DEFAULT NULL,
  `sample_epoch_p4` bigint(20) unsigned NOT NULL DEFAULT 0,
  `id_ts_file` int(11) NOT NULL,
  `last_date_listener` datetime DEFAULT current_timestamp(),
  `last_sample_epoch_listener` bigint(20) unsigned NOT NULL DEFAULT 0,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id_mysql_server_2` (`id_mysql_server`,`id_ts_file`),
  UNIQUE KEY `id_mysql_server` (`id_mysql_server`,`id_ts_file`,`date`),
  UNIQUE KEY `date` (`date`,`id_ts_file`,`id_mysql_server`),
  KEY `id_ts_file` (`id_ts_file`),
  KEY `id_mysql_server_4` (`id_mysql_server`,`date`),
  KEY `date_2` (`date`,`last_date_listener`),
  CONSTRAINT `ts_max_date_ibfk_3` FOREIGN KEY (`id_ts_file`) REFERENCES `ts_file` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=22859 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

SHOW CREATE TABLE mysql_server;

```
CREATE TABLE `mysql_server` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `is_deleted` tinyint(1) NOT NULL DEFAULT 0,
  `created_at` datetime NOT NULL DEFAULT current_timestamp(),
  `updated_at` datetime NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp(),
  `id_client` int(11) NOT NULL,
  `id_environment` int(11) NOT NULL,
  `name` varchar(100) CHARACTER SET utf8mb3 COLLATE utf8mb3_general_ci NOT NULL,
  `display_name` varchar(100) NOT NULL,
  `ip` varchar(255) CHARACTER SET latin1 COLLATE latin1_bin NOT NULL,
  `hostname` varchar(200) NOT NULL DEFAULT '',
  `login` varchar(32) CHARACTER SET utf8mb3 COLLATE utf8mb3_general_ci NOT NULL,
  `passwd` varchar(255) CHARACTER SET latin1 COLLATE latin1_bin NOT NULL,
  `database` varchar(64) NOT NULL,
  `is_password_crypted` int(11) NOT NULL,
  `is_ssl` int(11) NOT NULL DEFAULT 0,
  `port` int(11) NOT NULL,
  `ssh_nat` varchar(64) CHARACTER SET latin1 COLLATE latin1_bin NOT NULL DEFAULT '',
  `ssh_port` int(11) NOT NULL DEFAULT 22,
  `ssh_login` text CHARACTER SET latin1 COLLATE latin1_bin NOT NULL DEFAULT '\'\'',
  `is_sudo` int(11) NOT NULL DEFAULT 0,
  `is_root` int(11) NOT NULL DEFAULT 1,
  `is_monitored` int(11) NOT NULL DEFAULT 1,
  `monitoring_disabled_reason` enum('manual','license_quota') DEFAULT NULL,
  `is_proxy` int(11) NOT NULL DEFAULT 0,
  `is_vip` tinyint(1) NOT NULL DEFAULT 0,
  `is_binlog_relay` tinyint(1) NOT NULL DEFAULT 0,
  `is_clickhouse` tinyint(1) NOT NULL DEFAULT 0,
  `errant_gtid_set` text DEFAULT NULL COMMENT '#1280 P0.1 last computed errant GTID set
(canonical form, empty string = no errant)',
  `is_acknowledged` int(11) NOT NULL DEFAULT 0,
  `timeout` int(11) GENERATED ALWAYS AS (case when `is_proxy` = 1 then 11 else 1 end)
VIRTUAL,
  `errant_gtid_sampled_at` datetime DEFAULT NULL COMMENT '#1280 P0.1 wall-clock of the last
successful errant-set computation',
  `errant_gtid_ignore` tinyint(1) NOT NULL DEFAULT 0,
  `replica_lag_sla_seconds` int(11) NOT NULL DEFAULT 30,
  `is_out_of_service` tinyint(1) NOT NULL DEFAULT 0 COMMENT 'operator-flagged HS for reload
from master',
  PRIMARY KEY (`id`),
  UNIQUE KEY `name` (`name`),
```

```
  UNIQUE KEY `ip` (`ip`,`port`),
  KEY `id_client` (`id_client`),
  KEY `id_environment` (`id_environment`),
  KEY `is_monitored` (`id_client`,`is_monitored`,`is_deleted`) USING BTREE,
  KEY `is_deleted` (`is_deleted`)
) ENGINE=InnoDB AUTO_INCREMENT=337 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci WITH
SYSTEM VERSIONING
 PARTITION BY SYSTEM_TIME
PARTITIONS 2;
```

SHOW CREATE TABLE ts_value_general_int;

```
CREATE TABLE `ts_value_general_int` (
  `date` datetime NOT NULL,
  `id_ts_variable` int(11) NOT NULL,
  `id_mysql_server` int(11) NOT NULL,
  `value` bigint(20) unsigned DEFAULT NULL,
  PRIMARY KEY (`date`,`id_ts_variable`,`id_mysql_server`),
  KEY `id_mysql_server` (`id_mysql_server`,`id_ts_variable`,`date`)
) ENGINE=ROCKSDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci
 PARTITION BY RANGE (to_days(`date`))
(PARTITION `p740279` VALUES LESS THAN (740279) ENGINE = ROCKSDB,
 PARTITION `p740280` VALUES LESS THAN (740280) ENGINE = ROCKSDB,
 PARTITION `p740281` VALUES LESS THAN (740281) ENGINE = ROCKSDB,
 PARTITION `p740282` VALUES LESS THAN (740282) ENGINE = ROCKSDB,
 PARTITION `p740283` VALUES LESS THAN (740283) ENGINE = ROCKSDB,
 PARTITION `p740284` VALUES LESS THAN (740284) ENGINE = ROCKSDB,
 PARTITION `p740285` VALUES LESS THAN (740285) ENGINE = ROCKSDB,
 PARTITION `p740286` VALUES LESS THAN (740286) ENGINE = ROCKSDB,
 PARTITION `p740287` VALUES LESS THAN (740287) ENGINE = ROCKSDB,
 PARTITION `p740288` VALUES LESS THAN (740288) ENGINE = ROCKSDB,
 PARTITION `p740289` VALUES LESS THAN (740289) ENGINE = ROCKSDB,
 PARTITION `p740290` VALUES LESS THAN (740290) ENGINE = ROCKSDB,
 PARTITION `p740291` VALUES LESS THAN (740291) ENGINE = ROCKSDB,
 PARTITION `p740292` VALUES LESS THAN (740292) ENGINE = ROCKSDB,
 PARTITION `p740293` VALUES LESS THAN (740293) ENGINE = ROCKSDB,
 PARTITION `p740294` VALUES LESS THAN (740294) ENGINE = ROCKSDB,
 PARTITION `p740295` VALUES LESS THAN (740295) ENGINE = ROCKSDB,
 PARTITION `p740296` VALUES LESS THAN (740296) ENGINE = ROCKSDB,
 PARTITION `p740297` VALUES LESS THAN (740297) ENGINE = ROCKSDB,
 PARTITION `p740298` VALUES LESS THAN (740298) ENGINE = ROCKSDB,
```

```
  PARTITION `p740299` VALUES LESS THAN (740299) ENGINE = ROCKSDB,
  PARTITION `p740300` VALUES LESS THAN (740300) ENGINE = ROCKSDB,
  PARTITION `p740301` VALUES LESS THAN (740301) ENGINE = ROCKSDB,
  PARTITION `p740302` VALUES LESS THAN (740302) ENGINE = ROCKSDB,
  PARTITION `p740303` VALUES LESS THAN (740303) ENGINE = ROCKSDB,
  PARTITION `p740304` VALUES LESS THAN (740304) ENGINE = ROCKSDB,
  PARTITION `p740305` VALUES LESS THAN (740305) ENGINE = ROCKSDB,
  PARTITION `p740306` VALUES LESS THAN (740306) ENGINE = ROCKSDB,
  PARTITION `p740307` VALUES LESS THAN (740307) ENGINE = ROCKSDB,
  PARTITION `p740308` VALUES LESS THAN (740308) ENGINE = ROCKSDB,
  PARTITION `p740309` VALUES LESS THAN (740309) ENGINE = ROCKSDB,
  PARTITION `p740310` VALUES LESS THAN (740310) ENGINE = ROCKSDB);
```

采集器做了什么修改

`ts_variable` 表 name 引名的 `name, from)`。不同来源可以使用同一个指标名称，但此处给定名称与来源的组合最多对应一行。写入端已经知道来源。原先的读取条件指定了名称、类型和 radical，却漏掉

```
SELECT t.value AS previous_value
FROM ts_max_date AS m
JOIN ts_file AS f ON f.id = m.id_ts_file
JOIN ts_variable AS v ON v.id_ts_file = f.id
  AND v.name = 'mysql_available'
  AND v.`from` = 'mysql_server'  /* + */
  AND v.type = 'INT' AND v.radical = 'general'
JOIN mysql_server AS s ON s.id = m.id_mysql_server
  AND s.is_deleted = 0
LEFT JOIN ts_value_general_int AS t
  ON t.id_mysql_server = m.id_mysql_server
 AND t.id_ts_variable = v.id
 AND t.date = m.date
WHERE m.id_mysql_server = ? AND f.file_name = 'mysql_server'
LIMIT 1;
```

带注 on 一在 n 里是 前缀的 s 位 rve!; 这条路径集器值是各自对应的来源。唯一索引在切换之前就已存在。

加入来源后，3,791 行中仅多出 32 个分组

我们在 2026 年 11 月 21 日巴黎时间 21:25 对 variable 一个 3,791 行进行了分组。下表中的百分比为不同分组数除以表的总行数。

分组列	实际分组数	分组数 / 行数	SHOW INDEX 更新前 → 更新后

(`name`)	3,759	99.16%	3,841 → 3,840
(`name`, `from`)	3,791	100%	3,841 → 3,840
(`from`)	42	1.11%	42 → 42

`name` `from` **3,841** MariaDB

3,729 28 2 `name` (`from`) **32 99.16%** **100% 84** `mysql_available`

```
SELECT rows_per_name, COUNT(*) AS number_of_names
FROM (
    SELECT name, COUNT(*) AS rows_per_name
    FROM ts_variable
    GROUP BY name
) AS names
GROUP BY rows_per_name
ORDER BY rows_per_name;
```

`ts_variable`

21:00 `ANALYZE NO_WRITE_TO_BINLOG TABLE pmacontrol.ts_variable` OK `SHOW INDEX` **3,840**

InnoDB `mysql.innodb_index_stats` `name` `n_diff_pfx01` 3,757 **3,759** `n_diff_pfx02` 3,789 **3,791**

`ts_variable` `EXPLAIN` `ref` `const`

SQL ZIP

`name` `ref` v MariaDB 42 ref 168 const

`ts_file` (f)	1 · `const` · `file_name` · 130 B	1 · `const` · `file_name` · 130 B
`ts_max_date` (m)	2 · `const` · `id_mysql_server_2` · 8 B	3 · `const` · `id_mysql_server_2` · 8 B
`ts_variable` (v)	3 · `ref` · `name` · 102 B	2 · `const` · `name` · 168 B

表	修复前	修复后
`mysql_server` (s)	4 · `ref` · `PRIMARY` · 4 B	5 · `ref` · `PRIMARY` · 4 B
`ts_value_general_int` (t)	5 · `eq_ref` · `PRIMARY` · 13 B	4 · `const` · `PRIMARY` · 13 B

[illegible] MariaDB [illegible] `ts_value_general_int` [illegible] 32 [illegible]

[illegible] · [illegible] CSV [illegible] CSV。

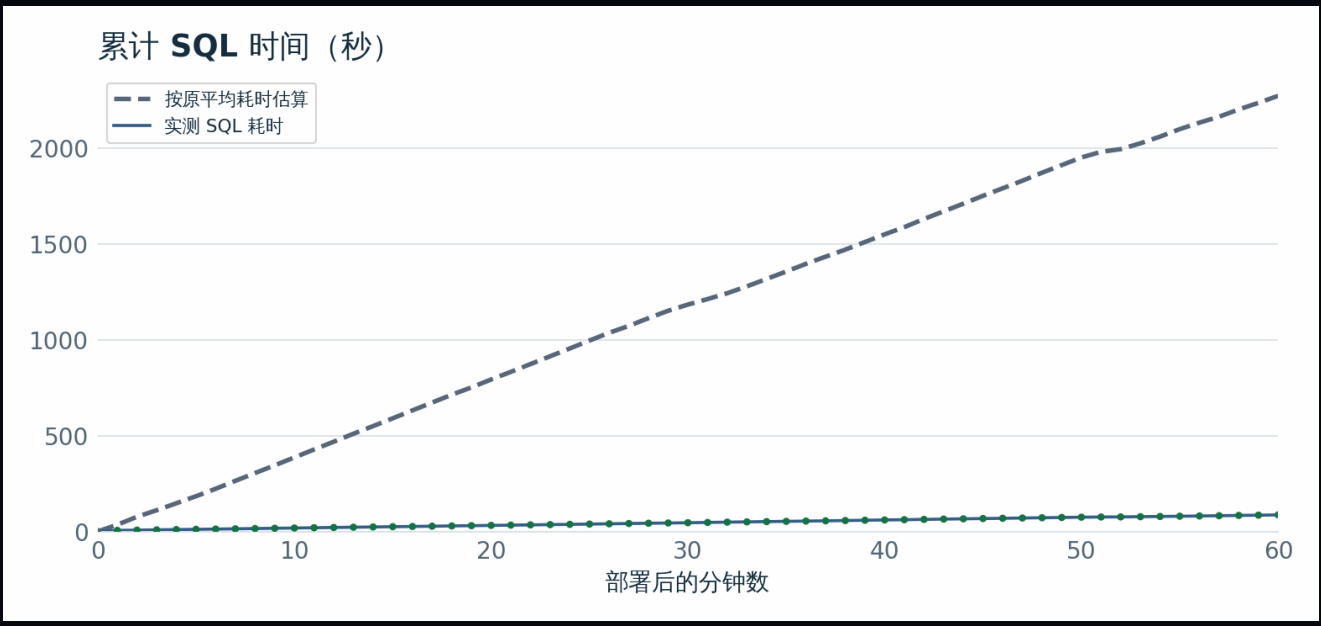
[illegible]

查询 · 50 [illegible]	[illegible]	[illegible]
[illegible]	35.085 ms	✓ **0.261 ms (÷134.27)**
[illegible]	0.107 ms	0.011 ms
`Statistics` [illegible]	54.983 ms	✓ **0.224 ms (÷245.46)**

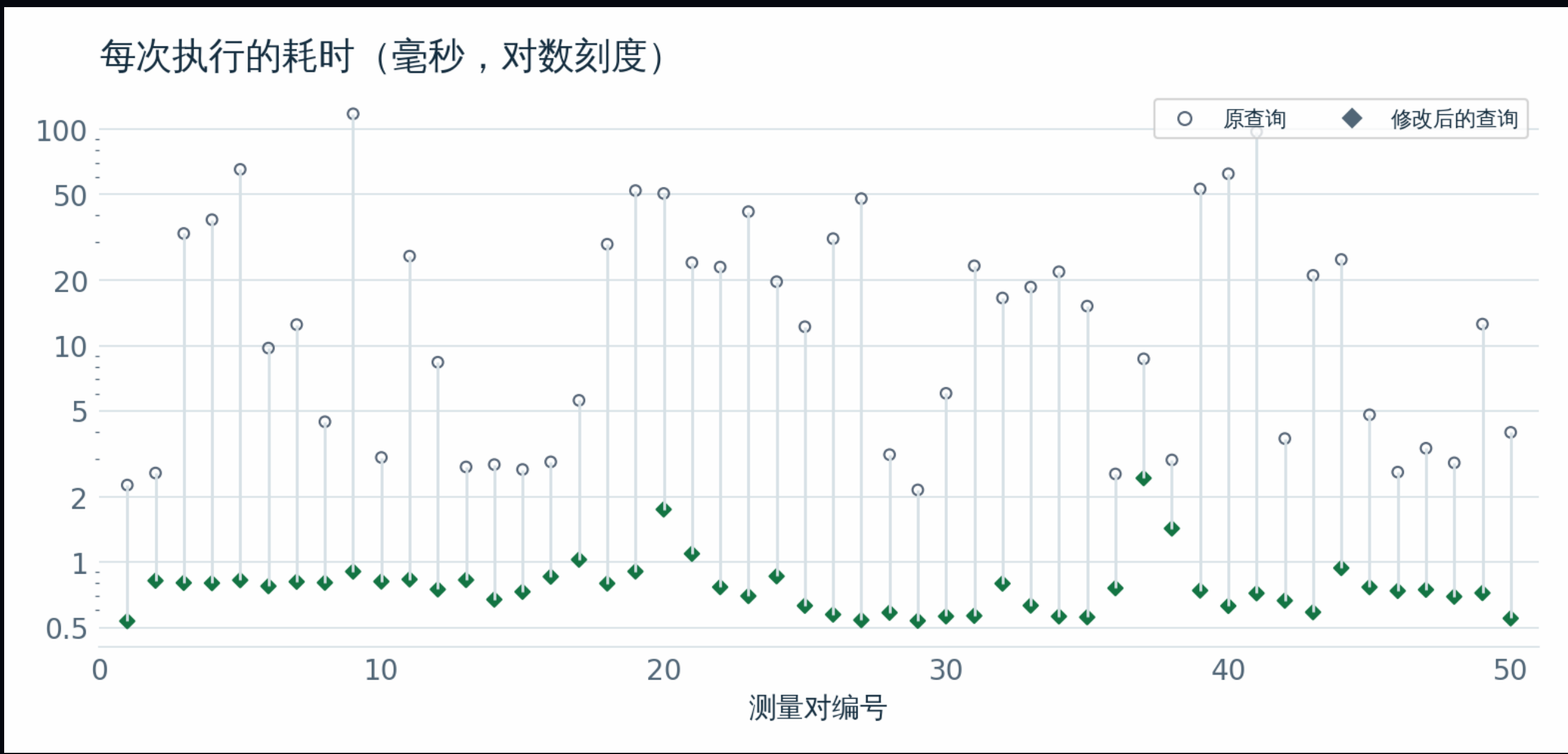
[illegible] `ANALYZE FORMAT=JSON` `$statistics` `SHOW PROFILE` [illegible]

[illegible] `name` [illegible] MariaDB [illegible] `name` [illegible] `from` [illegible] (`v.id` [illegible] 50 MariaDB 35.085 0.107 0.261 0.011 [illegible]

[illegible] SQL [illegible]



[illegible] **86.8 [illegible] 22 71.4 [illegible] 36 [illegible] 25 [illegible] SQL** [illegible] SQL [illegible] CPU [illegible]



在 50 台服务器上做配对测试

在关闭查询缓存的 MariaDB 11.8.10 上，我们从 111 个能返回有[illegible]服务器的[illegible]抽取 50 [illegible]每一对都是捕获到[illegible]一个只读事务里读取同一台服务器。25 对先执行原查询，其余 25

版[illegible]	最小值 (ms)	中位数 (ms)	平均值 (ms)	最大值 (ms)
修[illegible]前	2.159	12.543	21.686	117.530
修[illegible]	✓ 0.535 (÷4.03)	✓ 0.755 (÷16.62)	✓ 0.804 (÷26.98)	✓ 2.446 (÷48.05)

每一列连接同一台服务器的两次计时。对数刻度既能显示不足 1 毫秒的值，也能显示 117.53 毫秒的最大值。样本平均耗时之比为 26.98，两小时生产数据的加权平均耗时之比为 26.25。

与已部署查询比较的其他写法

每一行比较同一次实验中两种写法在相同服务器上的平均耗时。每组覆盖 50 台服务器，括号中是每种写法的执行次数。

查询写法	基准（毫秒）	本写法（毫秒）
仅名称 (150)	0.651	✗ 36.782 (×56.50)
仅名称，强制值表使用主键 (150)	0.651	✗ 1.674 (×2.57)
完整键，强制值表使用主键 (150)	0.651	0.630
强制日期表使用覆盖索引 (50)	0.692	✗ 20.232 (×29.25)
用 `EXISTS` 检查服务器是否有效 (150)	0.651	0.663
标量子查询读取值 (150)	0.617	0.546

查询写法	基准（毫秒）	本写法（毫秒）
固定连接顺序 (50)	0.692	✗ 41.737 (×60.34)
标量子查询 TS (150)	0.617	0.563
去 LIMIT 的标量子查询 (150)	0.617	0.560
去 LIMIT 的已部署写法 (150)	0.617	0.602
仅使用名称的标量子查询 (150)	0.617	1.218

匿名化查询与阶段测量数据（ZIP）

如何复算图表

观察窗口是巴黎时间 16:26:47 至 18:26:47。在各分段边界，用前后两次快照插值得到计数器值，部署时直接读取的调用计数用于校准切换点。部署前新摘要的 496 次测试调用以及部署后旧摘要的最后 224 次

每 COUNT_STAR 增 SUM 该 TIM 调 次 WA 算平均 COUNT_STAR 的增量，再除以十亿，把皮秒换算成毫秒。每小时的均值使用总时间和总调用数计算。与两小时的线上观测相比，配对测试能更好地隔离 SQL 修

下载数据与执行计划

- 计数器快照 (CSV)
- 每分钟序列与累计值 (CSV)
- 50 对匿名测量数据 (CSV)
- 详细执行计划 (CSV)
- MariaDB EXPLAIN 文档