# performance **schema**学习总结

## 版本

Version:5.6.21

## 一、编译

使用源码时cmake需要指定如下参数

-DWITH\_PERFSCHEMA\_STORAGE\_ENGINE=1

## 二、使用总结

performance schema主要是根据who/what/when/where/how这个五个要点来组织信息的

who：对哪些人进行监控。通过表setup\_actors进行配置

what：对哪些表进行监控。通过表setup\_objects进行配置

when：在什么时间点进行监控。通过表setup\_instruments进行配置

where：在哪里进行记录。通过表setup\_consumer进行配置

how：以什么方式进行记录。通过表setup\_timers进行配置

### setup表

#### setup\_actors

指定主机/用户/角色，被指定的主机/用户/角色所产生的事件将被记录。

建表语句

CREATE TABLE `setup\_actors` (

`HOST` char(60) CHARACTER SET utf8 COLLATE utf8\_bin NOT NULL DEFAULT '%',

`USER` char(16) CHARACTER SET utf8 COLLATE utf8\_bin NOT NULL DEFAULT '%',

`ROLE` char(16) CHARACTER SET utf8 COLLATE utf8\_bin NOT NULL DEFAULT '%'

) ENGINE=PERFORMANCE\_SCHEMA DEFAULT CHARSET=utf8

|  |  |
| --- | --- |
| 字段 | 说明 |
| Host | 指定跟踪的主机，来自该主机的操作将被记录 |
| user | 指定跟踪的用户，来自该用户的操作将被记录 |
| role | 指定跟踪的角色，来自该用户的操作将被记录，并不常用 |

说明

说明

1、记录之间是“或者”（or）的关系，只要满足setup\_actor表中任何一条记录的标准，将会视为满足要求

记录字段之间是“且”（and），同时满足的记录标准的，将会是为满足要求

2、通过在该表中增加/修改/删除记录，进行连接的监控配置，但请注意，修改后的配置信息对已有的连接无影响，对修改后新建立的连接产生影响

例子

1、监控来自任何主机的用户user1的事件，执行如下语句

insert into setup\_actors(HOST, USER,ROLE) values ('%', 'user1', '%');

#### setup\_objects

指定表/模式（schema），被指定表/模式（schema）上所产生的事件将被记录

建表语句

CREATE TABLE `setup\_objects` (

`OBJECT\_TYPE` enum('TABLE') NOT NULL DEFAULT 'TABLE',

`OBJECT\_SCHEMA` varchar(64) DEFAULT '%',

`OBJECT\_NAME` varchar(64) NOT NULL DEFAULT '%',

`ENABLED` enum('YES','NO') NOT NULL DEFAULT 'YES',

`TIMED` enum('YES','NO') NOT NULL DEFAULT 'YES'

) ENGINE=PERFORMANCE\_SCHEMA DEFAULT CHARSET=utf8

|  |  |
| --- | --- |
| 字段 | 说明 |
| OBJECT\_TYPE | 仅支持表类型的对象监控 |
| OBJECT\_SCHEMA | 表所在的schema，也就是database |
| OBJECT\_NAME | 被监控的对象名 |
| ENABLED | 是否开启该对象的监控 |
| TIMED | 待确定 |

说明

1、对setup\_ojbect的修改会立刻影响到被监控对象

例子

1、监控模式schema1下的表table1上所发生的事件

insert into setup\_objects(OBJECT\_SCHEMA, OBJECT\_NAME) values ('schema1', 'table1');

#### setup\_instruments

指定记录instruments。instruments也可以理解为事件，事件总是和时间相关，所以前面的纲领用when来概括。比如 “statement/sql/select”这个instrument，如果被开启了，则当select语句被执行时，将会记录下来作为参考。

建表语句

CREATE TABLE `setup\_instruments` (

`NAME` varchar(128) NOT NULL,

`ENABLED` enum('YES','NO') NOT NULL,

`TIMED` enum('YES','NO') NOT NULL

) ENGINE=PERFORMANCE\_SCHEMA DEFAULT CHARSET=utf8

|  |  |
| --- | --- |
| 字段 | 说明 |
| NAME | 进行监控事件，是类似“statement/sql/show\_databases”的内容 |
| ENABLED | 是否开启 |
| TIMED | 是否对事件进行计时 |

说明

1、对该表的更新将立刻生效

例子

一、监控select语句的执行

1、在setup\_instruments中打开对select的监控

update setup\_instruments set ENABLED='YES', TIMED='YES' where NAME='statement/sql/select';

2、执行select语句

select \* from t1;

3、在表events\_statements\_current中查看跟踪结果

红框部分和timed字段相关，如果设置为’NO’的情况，红框字段中的数据将为NULL。图1是设置为计时的结果，图2是设置为不计时的结果

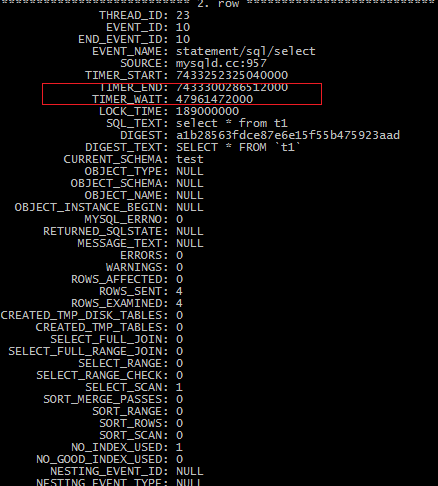


图1

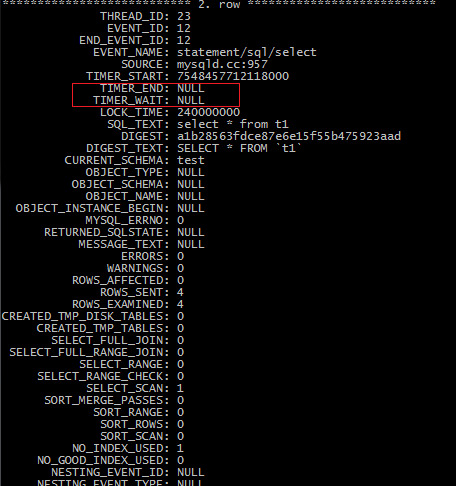


图2

#### setup\_consumers

指定记录instrument发生的表

建表语句

CREATE TABLE `setup\_consumers` (

`NAME` varchar(64) NOT NULL,

`ENABLED` enum('YES','NO') NOT NULL

) ENGINE=PERFORMANCE\_SCHEMA DEFAULT CHARSET=utf8

|  |  |
| --- | --- |
| 字段 | 说明 |
| NAME | 监控事件要保存的位置 |
| ENABLED | 是否开启 |

说明

1、对该表的修改将立即生效

2、NAME实际指的表名，如该字段的值为“events\_statements\_current”，则instruments下statement类型的事件将被保存到表“events\_statements\_current”中

3、NAME中的“global\_instrumentation/thread\_instrumentation/statements\_digest”，分别对应summary\_global/summary\_by\_thread/summary\_by\_digest表

例子

1、将发生的instrument记录到历史表中

update setup\_consumers set ENABLED='YES' where NAME='events\_statements\_history';

#### setup\_timers

指定记录instrument运行时间的记时方式

建表语句

CREATE TABLE `setup\_timers` (

`NAME` varchar(64) NOT NULL,

`TIMER\_NAME` enum('CYCLE','NANOSECOND','MICROSECOND','MILLISECOND','TICK') NOT NULL

) ENGINE=PERFORMANCE\_SCHEMA DEFAULT CHARSET=utf8

|  |  |
| --- | --- |
| 字段 | 说明 |
| NAME | 监控的事件类别 |
| TIMER\_NAME | 通过performance\_timers可以知道有哪些timer |

说明：

1、对该表的修改将立刻生效

2、只针对idle/wait/stage/statement的大类进行计时器的设置

例子

1、以纳秒方式记录instrument的statement分类

update setup\_timers set TIMER\_NAME='NANOSECOND' where NAME='statement';

### instance表

#### cond\_instances

建表语句

CREATE TABLE `cond\_instances` (

`NAME` varchar(128) NOT NULL,

`OBJECT\_INSTANCE\_BEGIN` bigint(20) unsigned NOT NULL

) ENGINE=PERFORMANCE\_SCHEMA DEFAULT CHARSET=utf8

|  |  |
| --- | --- |
| 字段 | 说明 |
| NAME | 待说明 |
| OBJECT\_INSTANCE\_BEGIN | 待说明 |

说明

1、在setup\_instruments中打开“wait/synch/cond” 分类下的开关，才会记录

#### file\_instances

建表语句

CREATE TABLE `file\_instances` (

`FILE\_NAME` varchar(512) NOT NULL,

`EVENT\_NAME` varchar(128) NOT NULL,

`OPEN\_COUNT` int(10) unsigned NOT NULL

) ENGINE=PERFORMANCE\_SCHEMA DEFAULT CHARSET=utf8

|  |  |
| --- | --- |
| 字段 | 说明 |
| FILE\_NAME | 文件名 |
| EVENT\_NAME | 打开文件的instrument |
| OPEN\_COUNT | 打开的次数。文件被打开并关闭，表示打开文件一次 |

说明

1、在setup\_instruments中打开“wait/io/file” 分类下的开关，才会记录

2、无法分辨哪些当前是打开的，当前是关闭的。所以在设计上，应该是open\_count和close\_count两个都存在，当open\_count=close\_count时，说明文件已经关闭。当open\_count>close\_count时，说明当前正在被打开。运行时，open\_count-close\_count一定要小于或者等于1.

3、如果没有被打开过的，则不在该表中有记录。

#### mutex\_instances

建表语句

CREATE TABLE `mutex\_instances` (

`NAME` varchar(128) NOT NULL,

`OBJECT\_INSTANCE\_BEGIN` bigint(20) unsigned NOT NULL,

`LOCKED\_BY\_THREAD\_ID` bigint(20) unsigned DEFAULT NULL

) ENGINE=PERFORMANCE\_SCHEMA DEFAULT CHARSET=utf8

|  |  |
| --- | --- |
| 字段 | 说明 |
| NAME |  |
| OBJECT\_INSTANCE\_BEGIN |  |
| LOCKED\_BY\_THREAD\_ID | 上锁的thread |

说明

1、在setup\_instruments中打开“wait/synch/mutex” 分类下的开关，才会记录

#### rwlock\_instances

CREATE TABLE `rwlock\_instances` (

`NAME` varchar(128) NOT NULL,

`OBJECT\_INSTANCE\_BEGIN` bigint(20) unsigned NOT NULL,

`WRITE\_LOCKED\_BY\_THREAD\_ID` bigint(20) unsigned DEFAULT NULL,

`READ\_LOCKED\_BY\_COUNT` int(10) unsigned NOT NULL

) ENGINE=PERFORMANCE\_SCHEMA DEFAULT CHARSET=utf8

|  |  |
| --- | --- |
| 字段 | 说明 |
| NAME | 事件名 |
| OBJECT\_INSTANCE\_BEGIN |  |
| WRITE\_LOCKED\_BY\_THREAD\_ID | 拥有该写锁的线程id，可通过show process list，或者performance\_schema的threads得到的 |
| READ\_LOCKED\_BY\_COUNT | 当前共享读锁的数量 |

说明

1、在setup\_instruments中打开“wait/synch/rwlock” 分类下的开关，才会记录

#### socket\_instances

建表语句

CREATE TABLE `socket\_instances` (

`EVENT\_NAME` varchar(128) NOT NULL,

`OBJECT\_INSTANCE\_BEGIN` bigint(20) unsigned NOT NULL,

`THREAD\_ID` bigint(20) unsigned DEFAULT NULL,

`SOCKET\_ID` int(11) NOT NULL,

`IP` varchar(64) NOT NULL,

`PORT` int(11) NOT NULL,

`STATE` enum('IDLE','ACTIVE') NOT NULL

) ENGINE=PERFORMANCE\_SCHEMA DEFAULT CHARSET=utf8

|  |  |
| --- | --- |
| 字段 | 说明 |
| EVENT\_NAME | 事件类型，只能是wait/io/socket/sql |
| OBJECT\_INSTANCE\_BEGIN | 待确定关联关系 |
| THREAD\_ID | 待确定关联关系 |
| SOCKET\_ID | 待确定关联关系 |
| IP | 端口ip地址 |
| PORT | 客户端的端口号 |
| STATE | 端口状态 |

说明：

1、在setup\_instruments中打开“wait/io/socket/sql”分类下的开关，才会记录

2、在setup\_instruments打开开关后，只针对新连接才会生效，对已有的连接不产生影响

3、在关闭开关后，已有的socket连接的状态也会因为操作发生变化

### connection表

#### accounts

建表语句

CREATE TABLE `accounts` (

`USER` char(16) CHARACTER SET utf8 COLLATE utf8\_bin DEFAULT NULL,

`HOST` char(60) CHARACTER SET utf8 COLLATE utf8\_bin DEFAULT NULL,

`CURRENT\_CONNECTIONS` bigint(20) NOT NULL,

`TOTAL\_CONNECTIONS` bigint(20) NOT NULL

) ENGINE=PERFORMANCE\_SCHEMA DEFAULT CHARSET=utf8

|  |  |
| --- | --- |
| 字段 | 说明 |
| USER | 用户名 |
| HOST | 用户所在主机 |
| CURRENT\_CONNECTIONS | 当前的连接数目 |
| TOTAL\_CONNECTIONS | 自服务器启动后，总共的连接次数 |

说明

1、记录中user和host的组合是唯一的

2、只要用户登录就会在此处留下信息，可用于监控用户的登录情况

#### hosts

建表语句

CREATE TABLE `hosts` (

`HOST` char(60) CHARACTER SET utf8 COLLATE utf8\_bin DEFAULT NULL,

`CURRENT\_CONNECTIONS` bigint(20) NOT NULL,

`TOTAL\_CONNECTIONS` bigint(20) NOT NULL

) ENGINE=PERFORMANCE\_SCHEMA DEFAULT CHARSET=utf8

|  |  |
| --- | --- |
| 字段 | 说明 |
| HOST | 主机名 |
| CURRENT\_CONNECTIONS | 当前的连接数目 |
| TOTAL\_CONNECTIONS | 自服务器启动后，总共的连接次数 |

说明：

1、统计host的连接情况

#### users

建表语句

CREATE TABLE `users` (

`USER` char(16) CHARACTER SET utf8 COLLATE utf8\_bin DEFAULT NULL,

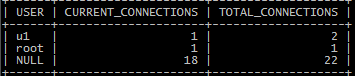
`CURRENT\_CONNECTIONS` bigint(20) NOT NULL,

`TOTAL\_CONNECTIONS` bigint(20) NOT NULL

) ENGINE=PERFORMANCE\_SCHEMA DEFAULT CHARSET=utf8

|  |  |
| --- | --- |
| 字段 | 说明 |
| USER | 用户名 |
| CURRENT\_CONNECTIONS | 当前使用该用户进行连接的数目 |
| TOTAL\_CONNECTIONS | 该用户总的连接数 |

内容类似如下



其中NULL是内部连接

### connection attribute

#### session\_account\_connect\_attrs

建表语句

CREATE TABLE `session\_account\_connect\_attrs` (

`PROCESSLIST\_ID` int(11) NOT NULL,

`ATTR\_NAME` varchar(32) COLLATE utf8\_bin NOT NULL,

`ATTR\_VALUE` varchar(1024) COLLATE utf8\_bin DEFAULT NULL,

`ORDINAL\_POSITION` int(11) DEFAULT NULL

) ENGINE=PERFORMANCE\_SCHEMA DEFAULT CHARSET=utf8 COLLATE=utf8\_bin

|  |  |
| --- | --- |
| 字段 | 说明 |
| PROCESSLIST\_ID | Show processlist中的id，用于定位sessio |
| ATTR\_NAME | 属性名 |
| ATTR\_VALUE | 属性值 |
| ORDINAL\_POSITION | 在属性集的顺序 |

说明：

1、显示当前会话的连接属性

#### session\_connect\_attrs

建表语句

CREATE TABLE `session\_connect\_attrs` (

`PROCESSLIST\_ID` int(11) NOT NULL,

`ATTR\_NAME` varchar(32) COLLATE utf8\_bin NOT NULL,

`ATTR\_VALUE` varchar(1024) COLLATE utf8\_bin DEFAULT NULL,

`ORDINAL\_POSITION` int(11) DEFAULT NULL

) ENGINE=PERFORMANCE\_SCHEMA DEFAULT CHARSET=utf8 COLLATE=utf8\_bin

|  |  |
| --- | --- |
| 字段 | 说明 |
| PROCESSLIST\_ID | Show processlist中的id，用于定位sessio |
| ATTR\_NAME | 属性名 |
| ATTR\_VALUE | 属性值 |
| ORDINAL\_POSITION | 在属性集的顺序 |

说明：

1、显示所有会话的连接属性

### instrument

#### instrument分类

wait/stage/statement是instrument的3个大类，该部分参考mysql手册针对每个instrument（事件）的说明，此处不进行说明

#### instrument保存

current/history/history long是保存instrument的3个大类

current：记录当前发生的实践，每个thread（session）只有一个记录

history：thread记录的事件的历史

history long：记录的内容与history相同，不同在于，当线程退出后该表中的数据不会将被删除，而history中该线程的中的数据将被删除

注1：在mysql的文档中，history long是在线程退出后清除该线程记录，但实测的情况和文档相反

##### events wait表

events\_waits\_current/events\_waits\_history/events\_waits\_history\_long

##### events stage表

##### events statement表

##### 说明

1、使用这些表进行信息记录需要预先进行setup\_instruments和setup\_consumers的设置

更新setup\_instruments/setup\_consumers中wait分类的记录信息，告知进行相应的wait分类记录；stage分类和statement分类同样操作

### 其他表说明

#### performance\_timers

performance schema的不同timer表示不同的计时方式，通过查询performance\_timer来获取timer的信息

建表语句

CREATE TABLE `performance\_timers` (

`TIMER\_NAME` enum('CYCLE','NANOSECOND','MICROSECOND','MILLISECOND','TICK') NOT NULL,

`TIMER\_FREQUENCY` bigint(20) DEFAULT NULL,

`TIMER\_RESOLUTION` bigint(20) DEFAULT NULL,

`TIMER\_OVERHEAD` bigint(20) DEFAULT NULL

) ENGINE=PERFORMANCE\_SCHEMA DEFAULT CHARSET=utf8

|  |  |
| --- | --- |
| 字段 | 说明 |
| TIMER\_NAME | timer的名字，用于区分不同类型的timer |
| TIMER\_FREQUENCY | 1秒钟内的计时次数 |
| TIMER\_RESOLUTION | 计时的分辨率。如果分辨率为10，得到的计时结果为10，则实际的计时结果为100 |
| TIMER\_OVERHEAD | timer本身支出的时间 |

计时类型

|  |  |
| --- | --- |
| Timer name | 说明 |
| CYCLE | 基于CPU时序的计时 |
| NANOSECOND | 基于纳秒的计时 |
| MICROSECOND | 基于微秒的计时 |
| MILLISECOND | 基于毫秒的计时 |
| TICK | 基于tick时序的计时 |

#### threads

建表语句

CREATE TABLE `threads` (

`THREAD\_ID` bigint(20) unsigned NOT NULL,

`NAME` varchar(128) NOT NULL,

`TYPE` varchar(10) NOT NULL,

`PROCESSLIST\_ID` bigint(20) unsigned DEFAULT NULL,

`PROCESSLIST\_USER` varchar(16) DEFAULT NULL,

`PROCESSLIST\_HOST` varchar(60) DEFAULT NULL,

`PROCESSLIST\_DB` varchar(64) DEFAULT NULL,

`PROCESSLIST\_COMMAND` varchar(16) DEFAULT NULL,

`PROCESSLIST\_TIME` bigint(20) DEFAULT NULL,

`PROCESSLIST\_STATE` varchar(64) DEFAULT NULL,

`PROCESSLIST\_INFO` longtext,

`PARENT\_THREAD\_ID` bigint(20) unsigned DEFAULT NULL,

`ROLE` varchar(64) DEFAULT NULL,

`INSTRUMENTED` enum('YES','NO') NOT NULL

) ENGINE=PERFORMANCE\_SCHEMA DEFAULT CHARSET=utf8

|  |  |
| --- | --- |
| 字段 | 说明 |
| THREAD\_ID | 线程id |
| NAME |  |
| TYPE | 类型，分为前端线程和后端线程。前端线程是用户连接线程，后端线程是服务线程。从mysql的实现看，后端线程 |
| PROCESSLIST\_ID | show processlist中显示的id |
| PROCESSLIST\_USER | 用户，仅对前端线程有效 |
| PROCESSLIST\_HOST | 主机，仅对前端线程有效 |
| PROCESSLIST\_DB | db，仅对前端线程有效 |
| PROCESSLIST\_COMMAND | 当前执行的命令，仅对前端线程有效 |
| PROCESSLIST\_TIME | 以秒计时，线程在当前状态的时间 |
| PROCESSLIST\_STATE | 线程的当前状态 |
| PROCESSLIST\_INFO | 线程当前信息 |
| PARENT\_THREAD\_ID | 父线程id，如果该进程是其他线程 |
| ROLE | 暂无意义 |
| INSTRUMENTED | 当前出发的instrumnet，参看instrument相关的介绍 |

说明

1、用于监控线程情况，其内容比show processlist更丰富

#### host\_cache

建表语句

CREATE TABLE `host\_cache` (

`IP` varchar(64) NOT NULL,

`HOST` varchar(255) CHARACTER SET utf8 COLLATE utf8\_bin DEFAULT NULL,

`HOST\_VALIDATED` enum('YES','NO') NOT NULL,

`SUM\_CONNECT\_ERRORS` bigint(20) NOT NULL,

`COUNT\_HOST\_BLOCKED\_ERRORS` bigint(20) NOT NULL,

`COUNT\_NAMEINFO\_TRANSIENT\_ERRORS` bigint(20) NOT NULL,

`COUNT\_NAMEINFO\_PERMANENT\_ERRORS` bigint(20) NOT NULL,

`COUNT\_FORMAT\_ERRORS` bigint(20) NOT NULL,

`COUNT\_ADDRINFO\_TRANSIENT\_ERRORS` bigint(20) NOT NULL,

`COUNT\_ADDRINFO\_PERMANENT\_ERRORS` bigint(20) NOT NULL,

`COUNT\_FCRDNS\_ERRORS` bigint(20) NOT NULL,

`COUNT\_HOST\_ACL\_ERRORS` bigint(20) NOT NULL,

`COUNT\_NO\_AUTH\_PLUGIN\_ERRORS` bigint(20) NOT NULL,

`COUNT\_AUTH\_PLUGIN\_ERRORS` bigint(20) NOT NULL,

`COUNT\_HANDSHAKE\_ERRORS` bigint(20) NOT NULL,

`COUNT\_PROXY\_USER\_ERRORS` bigint(20) NOT NULL,

`COUNT\_PROXY\_USER\_ACL\_ERRORS` bigint(20) NOT NULL,

`COUNT\_AUTHENTICATION\_ERRORS` bigint(20) NOT NULL,

`COUNT\_SSL\_ERRORS` bigint(20) NOT NULL,

`COUNT\_MAX\_USER\_CONNECTIONS\_ERRORS` bigint(20) NOT NULL,

`COUNT\_MAX\_USER\_CONNECTIONS\_PER\_HOUR\_ERRORS` bigint(20) NOT NULL,

`COUNT\_DEFAULT\_DATABASE\_ERRORS` bigint(20) NOT NULL,

`COUNT\_INIT\_CONNECT\_ERRORS` bigint(20) NOT NULL,

`COUNT\_LOCAL\_ERRORS` bigint(20) NOT NULL,

`COUNT\_UNKNOWN\_ERRORS` bigint(20) NOT NULL,

`FIRST\_SEEN` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',

`LAST\_SEEN` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',

`FIRST\_ERROR\_SEEN` timestamp NULL DEFAULT '0000-00-00 00:00:00',

`LAST\_ERROR\_SEEN` timestamp NULL DEFAULT '0000-00-00 00:00:00'

) ENGINE=PERFORMANCE\_SCHEMA DEFAULT CHARSET=utf8

用处不多，暂时不研究

|  |  |
| --- | --- |
| 字段 | 说明 |
|  |  |

说明

1、用于记录客户端主机和ip信息，避免DNS的查找

#### File I/O Summary Tables

##### file\_summary\_by\_event\_name

建表语句

CREATE TABLE `file\_summary\_by\_event\_name` (

`EVENT\_NAME` varchar(128) NOT NULL,

`COUNT\_STAR` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_WAIT` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_WAIT` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_WAIT` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_WAIT` bigint(20) unsigned NOT NULL,

`COUNT\_READ` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_READ` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_READ` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_READ` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_READ` bigint(20) unsigned NOT NULL,

`SUM\_NUMBER\_OF\_BYTES\_READ` bigint(20) NOT NULL,

`COUNT\_WRITE` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_WRITE` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_WRITE` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_WRITE` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_WRITE` bigint(20) unsigned NOT NULL,

`SUM\_NUMBER\_OF\_BYTES\_WRITE` bigint(20) NOT NULL,

`COUNT\_MISC` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_MISC` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_MISC` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_MISC` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_MISC` bigint(20) unsigned NOT NULL

) ENGINE=PERFORMANCE\_SCHEMA DEFAULT CHARSET=utf8

|  |  |
| --- | --- |
| 字段 | 说明 |
| EVENT\_NAME | 打开文件的instrument |

##### file\_summary\_by\_instance

建表语句

CREATE TABLE `file\_summary\_by\_instance` (

`FILE\_NAME` varchar(512) NOT NULL,

`EVENT\_NAME` varchar(128) NOT NULL,

`OBJECT\_INSTANCE\_BEGIN` bigint(20) unsigned NOT NULL,

`COUNT\_STAR` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_WAIT` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_WAIT` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_WAIT` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_WAIT` bigint(20) unsigned NOT NULL,

`COUNT\_READ` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_READ` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_READ` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_READ` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_READ` bigint(20) unsigned NOT NULL,

`SUM\_NUMBER\_OF\_BYTES\_READ` bigint(20) NOT NULL,

`COUNT\_WRITE` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_WRITE` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_WRITE` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_WRITE` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_WRITE` bigint(20) unsigned NOT NULL,

`SUM\_NUMBER\_OF\_BYTES\_WRITE` bigint(20) NOT NULL,

`COUNT\_MISC` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_MISC` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_MISC` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_MISC` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_MISC` bigint(20) unsigned NOT NULL

) ENGINE=PERFORMANCE\_SCHEMA DEFAULT CHARSET=utf8

|  |  |
| --- | --- |
| 字段 | 说明 |
| FILE\_NAME | 文件名 |
| EVENT\_NAME | 打开文件的instrument |

说明：

1、用于统计文件在某个instrument下的操作情况

##### 说明

1、by event name是按照event（也就是instrument）进行汇总，不同的文件在同一event下的操作信息被汇总到一起。

2、by instance是按照文件进行汇总（不知道是否按照event）

3、通过truncate命令清除该表的信息，不过不是删除行，而是将统计信息清零。

4、部分字段的说明

COUNT\_READ, SUM\_TIMER\_READ, MIN\_TIMER\_READ, AVG\_TIMER\_READ, MAX\_TIMER\_READ, SUM\_NUMBER\_OF\_BYTES\_READ

读操作的统计信息

COUNT\_WRITE, SUM\_TIMER\_WRITE, MIN\_TIMER\_WRITE, AVG\_TIMER\_WRITE, MAX\_TIMER\_WRITE, SUM\_NUMBER\_OF\_BYTES\_WRITE

写操作的统计信息

COUNT\_MISC, SUM\_TIMER\_MISC, MIN\_TIMER\_MISC, AVG\_TIMER\_MISC, MAX\_TIMER\_MISC

其他操作的统计信息

COUNT\_STAR, SUM\_TIMER\_WAIT, MIN\_TIMER\_WAIT, AVG\_TIMER\_WAIT, MAX\_TIMER\_WAIT

综合前三者的信息，如COUNT\_STAR= COUNT\_READ+ COUNT\_WRITE+ COUNT\_MISC

#### objects\_summary\_global\_by\_type

建表语句

CREATE TABLE `objects\_summary\_global\_by\_type` (

`OBJECT\_TYPE` varchar(64) DEFAULT NULL,

`OBJECT\_SCHEMA` varchar(64) DEFAULT NULL,

`OBJECT\_NAME` varchar(64) DEFAULT NULL,

`COUNT\_STAR` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_WAIT` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_WAIT` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_WAIT` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_WAIT` bigint(20) unsigned NOT NULL

) ENGINE=PERFORMANCE\_SCHEMA DEFAULT CHARSET=utf8

|  |  |
| --- | --- |
| 字段 | 说明 |
| OBJECT\_TYPE | 对象类型 |
| OBJECT\_SCHEMA | 对象所在schema |
| OBJECT\_NAME | 对象名 |
| COUNT\_STAR |  |
| SUM\_TIMER\_WAIT |  |
| MIN\_TIMER\_WAIT |  |
| AVG\_TIMER\_WAIT |  |
| MAX\_TIMER\_WAIT |  |

##### 待完成

#### socket summary表

##### socket\_summary\_by\_instance

建表语句

CREATE TABLE `socket\_summary\_by\_instance` (

`EVENT\_NAME` varchar(128) NOT NULL,

`OBJECT\_INSTANCE\_BEGIN` bigint(20) unsigned NOT NULL,

`COUNT\_STAR` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_WAIT` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_WAIT` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_WAIT` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_WAIT` bigint(20) unsigned NOT NULL,

`COUNT\_READ` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_READ` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_READ` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_READ` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_READ` bigint(20) unsigned NOT NULL,

`SUM\_NUMBER\_OF\_BYTES\_READ` bigint(20) unsigned NOT NULL,

`COUNT\_WRITE` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_WRITE` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_WRITE` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_WRITE` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_WRITE` bigint(20) unsigned NOT NULL,

`SUM\_NUMBER\_OF\_BYTES\_WRITE` bigint(20) unsigned NOT NULL,

`COUNT\_MISC` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_MISC` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_MISC` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_MISC` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_MISC` bigint(20) unsigned NOT NULL

) ENGINE=PERFORMANCE\_SCHEMA DEFAULT CHARSET=utf8

|  |  |
| --- | --- |
| 字段 | 说明 |
| EVENT\_NAME | Instrument类型 |
| OBJECT\_INSTANCE\_BEGIN | 标识socket对象，通过该对象可以在socket\_instances表中定位到thread |

说明

1、按照socket的instance进行统计，每产生一个连接，就会有一行记录

##### socket\_summary\_by\_event\_name

建表语句

CREATE TABLE `socket\_summary\_by\_event\_name` (

`EVENT\_NAME` varchar(128) NOT NULL,

`COUNT\_STAR` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_WAIT` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_WAIT` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_WAIT` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_WAIT` bigint(20) unsigned NOT NULL,

`COUNT\_READ` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_READ` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_READ` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_READ` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_READ` bigint(20) unsigned NOT NULL,

`SUM\_NUMBER\_OF\_BYTES\_READ` bigint(20) unsigned NOT NULL,

`COUNT\_WRITE` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_WRITE` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_WRITE` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_WRITE` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_WRITE` bigint(20) unsigned NOT NULL,

`SUM\_NUMBER\_OF\_BYTES\_WRITE` bigint(20) unsigned NOT NULL,

`COUNT\_MISC` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_MISC` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_MISC` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_MISC` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_MISC` bigint(20) unsigned NOT NULL

) ENGINE=PERFORMANCE\_SCHEMA DEFAULT CHARSET=utf8

|  |  |
| --- | --- |
| 字段 | 说明 |
| EVENT\_NAME | Instrument类型 |

说明

1、按照事件进行统计，也就是wait/io/socket/sql分类下，每类的情况统计，有多少个分类就有多少个记录。在当前版本下只有3个分类，所以只有3个记录。

##### 说明

1、用于统计socket的使用情况。与socket\_instances使用相同的开关打开监控

2、如下是summary中都存在的列

COUNT\_STAR, SUM\_TIMER\_WAIT, MIN\_TIMER\_WAIT, AVG\_TIMER\_WAIT, MAX\_TIMER\_WAIT

统计全部数据，如COUNT\_STAR= COUNT\_READ+ COUNT\_WRITE+ COUNT\_MISC

COUNT\_READ, SUM\_TIMER\_READ, MIN\_TIMER\_READ, AVG\_TIMER\_READ, MAX\_TIMER\_READ, SUM\_NUMBER\_OF\_BYTES\_READ

统计读

COUNT\_WRITE, SUM\_TIMER\_WRITE, MIN\_TIMER\_WRITE, AVG\_TIMER\_WRITE, MAX\_TIMER\_WRITE, SUM\_NUMBER\_OF\_BYTES\_WRITE

统计写

COUNT\_MISC, SUM\_TIMER\_MISC, MIN\_TIMER\_MISC, AVG\_TIMER\_MISC, MAX\_TIMER\_MISC

其他杂项统计

#### table summary表

##### table\_io\_waits\_summary\_by\_index\_usage

建表语句

CREATE TABLE `table\_io\_waits\_summary\_by\_index\_usage` (

`OBJECT\_TYPE` varchar(64) DEFAULT NULL,

`OBJECT\_SCHEMA` varchar(64) DEFAULT NULL,

`OBJECT\_NAME` varchar(64) DEFAULT NULL,

`INDEX\_NAME` varchar(64) DEFAULT NULL,

`COUNT\_STAR` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_WAIT` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_WAIT` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_WAIT` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_WAIT` bigint(20) unsigned NOT NULL,

`COUNT\_READ` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_READ` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_READ` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_READ` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_READ` bigint(20) unsigned NOT NULL,

`COUNT\_WRITE` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_WRITE` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_WRITE` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_WRITE` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_WRITE` bigint(20) unsigned NOT NULL,

`COUNT\_FETCH` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_FETCH` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_FETCH` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_FETCH` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_FETCH` bigint(20) unsigned NOT NULL,

`COUNT\_INSERT` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_INSERT` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_INSERT` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_INSERT` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_INSERT` bigint(20) unsigned NOT NULL,

`COUNT\_UPDATE` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_UPDATE` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_UPDATE` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_UPDATE` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_UPDATE` bigint(20) unsigned NOT NULL,

`COUNT\_DELETE` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_DELETE` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_DELETE` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_DELETE` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_DELETE` bigint(20) unsigned NOT NULL

) ENGINE=PERFORMANCE\_SCHEMA DEFAULT CHARSET=utf8

|  |  |
| --- | --- |
| 字段 | 说明 |
| OBJECT\_TYPE | 对象类型 |
| OBJECT\_SCHEMA | 对象schema |
| OBJECT\_NAME | 对象名 |
| INDEX\_NAME | 索引名 |

说明

1、每个索引为一条统计记录，一个表存在多个索引，该表则有多个记录

2、INDEX\_NAME字段为null，说明是主表，不存在索引名

##### table\_io\_waits\_summary\_by\_table

建表语句

CREATE TABLE `table\_io\_waits\_summary\_by\_table` (

`OBJECT\_TYPE` varchar(64) DEFAULT NULL,

`OBJECT\_SCHEMA` varchar(64) DEFAULT NULL,

`OBJECT\_NAME` varchar(64) DEFAULT NULL,

`COUNT\_STAR` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_WAIT` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_WAIT` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_WAIT` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_WAIT` bigint(20) unsigned NOT NULL,

`COUNT\_READ` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_READ` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_READ` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_READ` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_READ` bigint(20) unsigned NOT NULL,

`COUNT\_WRITE` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_WRITE` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_WRITE` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_WRITE` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_WRITE` bigint(20) unsigned NOT NULL,

`COUNT\_FETCH` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_FETCH` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_FETCH` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_FETCH` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_FETCH` bigint(20) unsigned NOT NULL,

`COUNT\_INSERT` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_INSERT` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_INSERT` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_INSERT` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_INSERT` bigint(20) unsigned NOT NULL,

`COUNT\_UPDATE` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_UPDATE` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_UPDATE` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_UPDATE` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_UPDATE` bigint(20) unsigned NOT NULL,

`COUNT\_DELETE` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_DELETE` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_DELETE` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_DELETE` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_DELETE` bigint(20) unsigned NOT NULL

) ENGINE=PERFORMANCE\_SCHEMA DEFAULT CHARSET=utf8

|  |  |
| --- | --- |
| 字段 | 说明 |
| OBJECT\_TYPE | 对象类型 |
| OBJECT\_SCHEMA | 对象schema |
| OBJECT\_NAME | 对象名 |

说明

1、每个表为一条统计记录

##### table\_lock\_waits\_summary\_by\_table

建表语句

CREATE TABLE `table\_lock\_waits\_summary\_by\_table` (

`OBJECT\_TYPE` varchar(64) DEFAULT NULL,

`OBJECT\_SCHEMA` varchar(64) DEFAULT NULL,

`OBJECT\_NAME` varchar(64) DEFAULT NULL,

`COUNT\_STAR` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_WAIT` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_WAIT` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_WAIT` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_WAIT` bigint(20) unsigned NOT NULL,

`COUNT\_READ` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_READ` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_READ` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_READ` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_READ` bigint(20) unsigned NOT NULL,

`COUNT\_WRITE` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_WRITE` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_WRITE` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_WRITE` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_WRITE` bigint(20) unsigned NOT NULL,

`COUNT\_READ\_NORMAL` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_READ\_NORMAL` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_READ\_NORMAL` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_READ\_NORMAL` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_READ\_NORMAL` bigint(20) unsigned NOT NULL,

`COUNT\_READ\_WITH\_SHARED\_LOCKS` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_READ\_WITH\_SHARED\_LOCKS` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_READ\_WITH\_SHARED\_LOCKS` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_READ\_WITH\_SHARED\_LOCKS` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_READ\_WITH\_SHARED\_LOCKS` bigint(20) unsigned NOT NULL,

`COUNT\_READ\_HIGH\_PRIORITY` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_READ\_HIGH\_PRIORITY` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_READ\_HIGH\_PRIORITY` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_READ\_HIGH\_PRIORITY` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_READ\_HIGH\_PRIORITY` bigint(20) unsigned NOT NULL,

`COUNT\_READ\_NO\_INSERT` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_READ\_NO\_INSERT` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_READ\_NO\_INSERT` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_READ\_NO\_INSERT` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_READ\_NO\_INSERT` bigint(20) unsigned NOT NULL,

`COUNT\_READ\_EXTERNAL` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_READ\_EXTERNAL` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_READ\_EXTERNAL` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_READ\_EXTERNAL` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_READ\_EXTERNAL` bigint(20) unsigned NOT NULL,

`COUNT\_WRITE\_ALLOW\_WRITE` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_WRITE\_ALLOW\_WRITE` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_WRITE\_ALLOW\_WRITE` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_WRITE\_ALLOW\_WRITE` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_WRITE\_ALLOW\_WRITE` bigint(20) unsigned NOT NULL,

`COUNT\_WRITE\_CONCURRENT\_INSERT` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_WRITE\_CONCURRENT\_INSERT` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_WRITE\_CONCURRENT\_INSERT` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_WRITE\_CONCURRENT\_INSERT` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_WRITE\_CONCURRENT\_INSERT` bigint(20) unsigned NOT NULL,

`COUNT\_WRITE\_DELAYED` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_WRITE\_DELAYED` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_WRITE\_DELAYED` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_WRITE\_DELAYED` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_WRITE\_DELAYED` bigint(20) unsigned NOT NULL,

`COUNT\_WRITE\_LOW\_PRIORITY` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_WRITE\_LOW\_PRIORITY` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_WRITE\_LOW\_PRIORITY` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_WRITE\_LOW\_PRIORITY` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_WRITE\_LOW\_PRIORITY` bigint(20) unsigned NOT NULL,

`COUNT\_WRITE\_NORMAL` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_WRITE\_NORMAL` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_WRITE\_NORMAL` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_WRITE\_NORMAL` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_WRITE\_NORMAL` bigint(20) unsigned NOT NULL,

`COUNT\_WRITE\_EXTERNAL` bigint(20) unsigned NOT NULL,

`SUM\_TIMER\_WRITE\_EXTERNAL` bigint(20) unsigned NOT NULL,

`MIN\_TIMER\_WRITE\_EXTERNAL` bigint(20) unsigned NOT NULL,

`AVG\_TIMER\_WRITE\_EXTERNAL` bigint(20) unsigned NOT NULL,

`MAX\_TIMER\_WRITE\_EXTERNAL` bigint(20) unsigned NOT NULL

) ENGINE=PERFORMANCE\_SCHEMA DEFAULT CHARSET=utf8

|  |  |
| --- | --- |
| 字段 | 说明 |
| OBJECT\_TYPE | 对象类型 |
| OBJECT\_SCHEMA | 对象schema |
| OBJECT\_NAME | 对象名 |

说明

1、每个表为一条统计记录

2、需设置setup\_instruments的“wait/lock/table/sql/handler”分类下的开关，进行记录

##### 说明

1、对table的io和lock进行统计

2、相同字段部分

COUNT\_STAR, SUM\_TIMER\_WAIT, MIN\_TIMER\_WAIT, AVG\_TIMER\_WAIT, MAX\_TIMER\_WAIT

COUNT\_READ, SUM\_TIMER\_READ, MIN\_TIMER\_READ, AVG\_TIMER\_READ, MAX\_TIMER\_READ

COUNT\_WRITE, SUM\_TIMER\_WRITE, MIN\_TIMER\_WRITE, AVG\_TIMER\_WRITE, MAX\_TIMER\_WRITE

COUNT\_READ\_NORMAL, SUM\_TIMER\_READ\_NORMAL, MIN\_TIMER\_READ\_NORMAL, AVG\_TIMER\_READ\_NORMAL, MAX\_TIMER\_READ\_NORMAL

COUNT\_READ\_WITH\_SHARED\_LOCKS, SUM\_TIMER\_READ\_WITH\_SHARED\_LOCKS, MIN\_TIMER\_READ\_WITH\_SHARED\_LOCKS, AVG\_TIMER\_READ\_WITH\_SHARED\_LOCKS, MAX\_TIMER\_READ\_WITH\_SHARED\_LOCKS

COUNT\_READ\_HIGH\_PRIORITY, SUM\_TIMER\_READ\_HIGH\_PRIORITY, MIN\_TIMER\_READ\_HIGH\_PRIORITY, AVG\_TIMER\_READ\_HIGH\_PRIORITY, MAX\_TIMER\_READ\_HIGH\_PRIORITY

COUNT\_READ\_NO\_INSERT, SUM\_TIMER\_READ\_NO\_INSERT, MIN\_TIMER\_READ\_NO\_INSERT, AVG\_TIMER\_READ\_NO\_INSERT, MAX\_TIMER\_READ\_NO\_INSERT

COUNT\_READ\_EXTERNAL, SUM\_TIMER\_READ\_EXTERNAL, MIN\_TIMER\_READ\_EXTERNAL, AVG\_TIMER\_READ\_EXTERNAL, MAX\_TIMER\_READ\_EXTERNAL

COUNT\_WRITE\_ALLOW\_WRITE, SUM\_TIMER\_WRITE\_ALLOW\_WRITE, MIN\_TIMER\_WRITE\_ALLOW\_WRITE, AVG\_TIMER\_WRITE\_ALLOW\_WRITE, MAX\_TIMER\_WRITE\_ALLOW\_WRITE

COUNT\_WRITE\_CONCURRENT\_INSERT, SUM\_TIMER\_WRITE\_CONCURRENT\_INSERT, MIN\_TIMER\_WRITE\_CONCURRENT\_INSERT, AVG\_TIMER\_WRITE\_CONCURRENT\_INSERT, MAX\_TIMER\_WRITE\_CONCURRENT\_INSERT

COUNT\_WRITE\_DELAYED, SUM\_TIMER\_WRITE\_DELAYED, MIN\_TIMER\_WRITE\_DELAYED, AVG\_TIMER\_WRITE\_DELAYED, MAX\_TIMER\_WRITE\_DELAYED

COUNT\_WRITE\_LOW\_PRIORITY, SUM\_TIMER\_WRITE\_LOW\_PRIORITY, MIN\_TIMER\_WRITE\_LOW\_PRIORITY, AVG\_TIMER\_WRITE\_LOW\_PRIORITY, MAX\_TIMER\_WRITE\_LOW\_PRIORITY

COUNT\_WRITE\_NORMAL, SUM\_TIMER\_WRITE\_NORMAL, MIN\_TIMER\_WRITE\_NORMAL, AVG\_TIMER\_WRITE\_NORMAL, MAX\_TIMER\_WRITE\_NORMAL

COUNT\_WRITE\_EXTERNAL, SUM\_TIMER\_WRITE\_EXTERNAL, MIN\_TIMER\_WRITE\_EXTERNAL, AVG\_TIMER\_WRITE\_EXTERNAL, MAX\_TIMER\_WRITE\_EXTERNAL

### 概念层级

下图所描绘为概念的从属关系

who

actor

what

object

when

instrument

where

consumer

how

timer

when

instrument

wait

stage

statement

where

consumer

current

history long

history

### 操作实例

1、仅对指定的用户进行性能监控

创建用户

CREATE USER u1 IDENTIFIED BY ‘123’;

GRANT ALL ON \*.\* TO u1;

修改配置表

truncate setup\_actors;

insert setup\_actors values (‘%’, ‘u1’, ‘%’);

2、定位mutex所引起线程阻塞

2-1、假设thread id为X的线程在等待mutex

2-2、定位thread等待的锁对象

SELECT OBJECT\_INSTANCE\_BEGIN FROM events\_waits\_current WHERE THREAD\_ID = X;

OBJECT\_INSTANCE\_BEGIN字段就是mutex对象的标识，记作M

2-3、通过mutex对象的标识M，获得拥有该mutex对象的线程

SELECT LOCKED\_BY\_THREAD\_ID FROM mutex\_instances WHERE OBJECT\_INSTANCE\_BEGIN = M;

LOCKED\_BY\_THREAD\_ID字段就是拥有该mutex对象的线程，记作Y

2-4、查看Y线程正在做的事情

SELECT \* FROM events\_waits\_current WHERE THREAD\_ID = Y;

待完成实例

查找读写字节最多的文件

读入多：select file\_name, SUM\_NUMBER\_OF\_BYTES\_READ from file\_summary\_by\_instance order by SUM\_NUMBER\_OF\_BYTES\_READ desc

写出多：使用SUM\_NUMBER\_OF\_BYTES\_WRITE字段排序

### 性能说明

暂无性能对比说明

## 代码总结

### 1、instrument

instrument分为statement/wait/stage类型

#### 1-1、statement

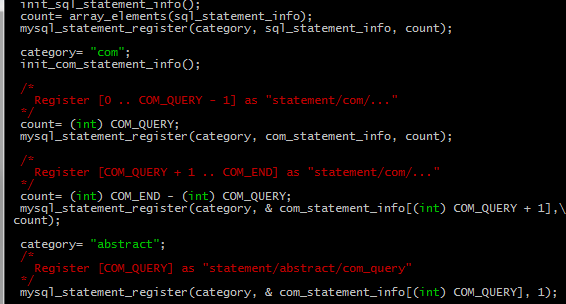


statement的instrument分为3类，sql、om和abstract分别用如下变量进行定义

sql\_statement\_info

com\_statement\_info

在函数init\_server\_psi\_keys中调用mysql\_statement\_register进行初始化



#### 1-2、wait

wait的instrument分为3类，io/lcok/synch，分别代表输入输出/所/同步

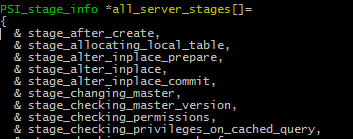
#### 1-3、stage

stage下只有一个sql子类

在init\_server\_psi\_keys函数中进行注册



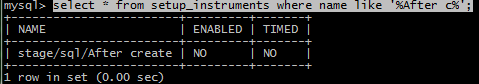
其中all\_server\_stages看上去是这样的，有若干变量组成



举stage\_after\_create



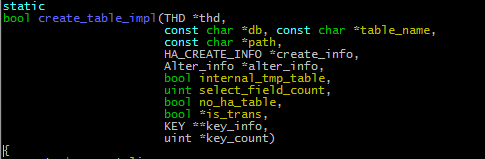
对应到performance\_schema中看起来是这样的



实际使用是这样的



在建表的函数中进行状态设置



show processlist中的State字段显示的也是通过THD\_STAGE\_INFO所注册的信息

#### instrument的注册

## 参考

http://mysqlintheenterprise.com/2013/03/21/a-visual-guide-to-the-mysql-performance-schema/

http://dev.mysql.com/doc/refman/5.6/en/performance-schema.html

mysql源码