性能优化专题

MYSQL的配置优化

主讲人：ROBERT: 2831742582

目录 /CONTENTS

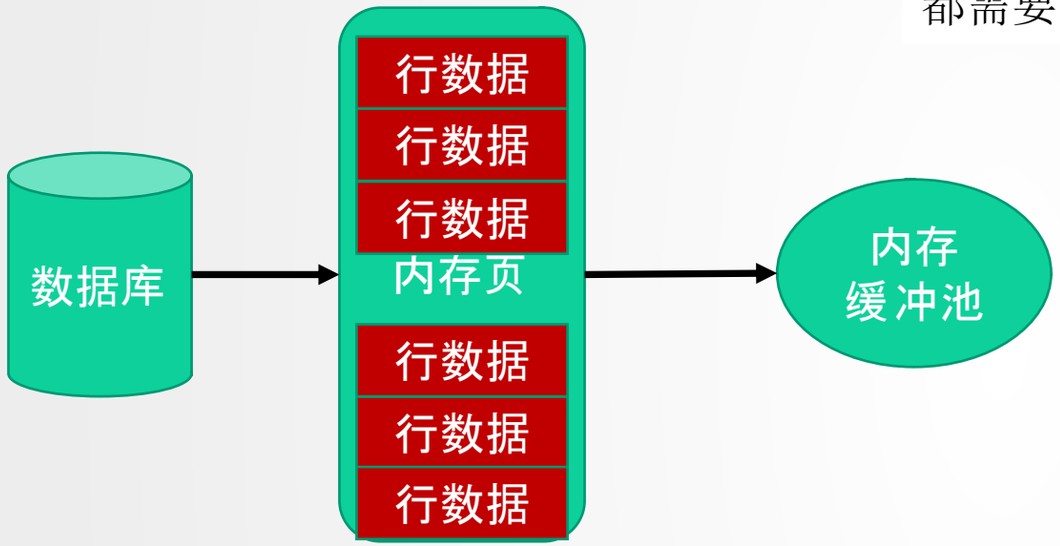
01

SQL优化复习

SQL

索引

从数据库加载一行或一批数据
随机读取



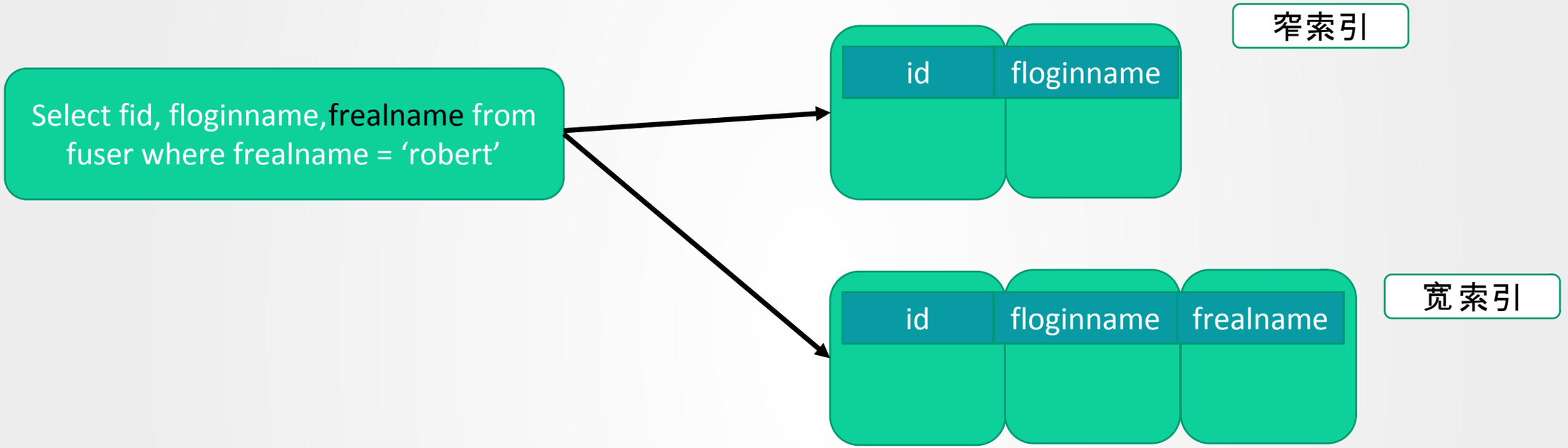
数据库等待一个页从磁盘读取到缓存池的所需要的成本巨大的，无论我们是想要读取一个页面上的多条数据还是一条数据，都需要消耗约 10ms 左右的时间：

索引

内存读取
最快

顺序读取
磁盘读取性能
机械盘可达40m/s

索引



(id, floginname) 就是一个窄索引，因为该索引没有包含存在于 SQL 查询中的 frealname 列，而 (id, floginname, frealname) 就是该查询的一个宽索引了，**它包含这个查询中所需要的全部数据列。**

索引

在单表索引中尽量设计出三星索引，如下描述：

1.第一颗星需要取出所有等值谓词中的列，作为索引开头的最开始的列（任意顺序）；
2.第二颗星需要将 ORDER BY 列加入索引中；
3.第三颗星需要将查询语句剩余的列全部加入到索引中；

索引顺序如下两种考虑：
1、最小化表扫描
2、避免排序

```
Select fid, floginname,frealname from fuser where frealname = 'robert' and fage between 20 and 30 order by fcreatetime desc
```

(floginname, frealname, fage, fcreatetime)

(floginname, frealname, fcreatetime , fage)

SQL优化顺序

主要原则

1、尽量少作计算
2、尽量少 join
3、尽量少排序
4、尽量避免 select *
5、尽量用 join 代替子查询
6、尽量少 or
7、尽量用 union all 代替 union
8、尽量早过滤
9、避免类型转换
10、优先优化高并发的 SQL，而不是执行频率低某些“大”SQL
11、从全局出发优化，而不是片面调整
12、尽可能对每一条运行在数据库中的SQL进行 explain

SQL优化顺序

子查询的错误用法

```
Select fid, fprize, famount, (select frealname from fuser where fid = fus_fid) from fentrust_vcoin where date(fcreatetime) = ‘2018-02-01’

Select e.fid, e.fprize, e.famount, u.frealname
From fentrust_vcoin e
Inner join fuser u on e.fus_fid = u.fid
Where date(e.fcreatetime) = ‘2018-02-01’
```

02

一些关键配置

CONFIG

配置文件的位置

MySQL配置文件

/etc/my.cnf　或者　/etc/my.cnf.d/server.cnf

几个关键的文件：
.pid文件，记录了进程id
.sock文件，是内部通信使用的socket接口，比3306快
.log文件，日志文件
.cnf或.conf文件，配置文件

安装目录：basedir
数据目录：datadir

在哪里保存你的数据

基本配置，指定数据目录

以下配置不要照抄，my.cnf或者server.cnf

```
[mysqld]
user = mysql
port = 3306
socket = /data/3306/mysql.sock，#这里指定了一个特别的连接
basedir = /usr/local/mysql
datadir = /data/3306/data

[client]
port = 3306
socket = /data/3306/mysql.sock，在客户端也要声明它，命令行要用到
```

查询缓存要不要开

写入频繁的数据库，不要开查询缓存

query_cache_size

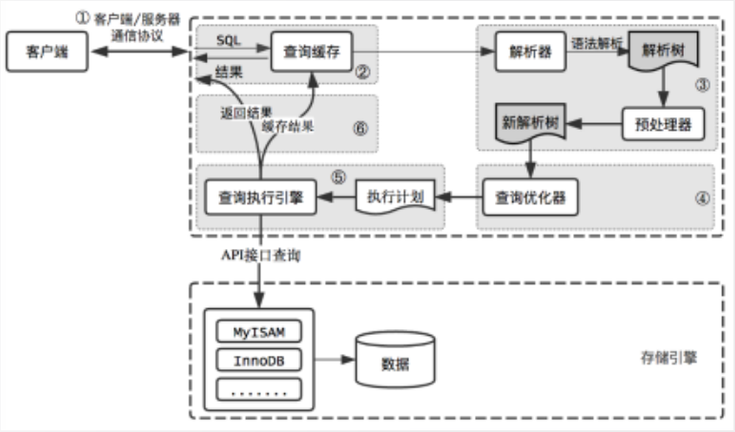
Query_cache里的数据又怎么处理呢？首先要把Query_cache和该表相关的语句全部置为失效，然后在写入更新。那么如果Query_cache非常大，该表的查询结构又比较多，查询语句失效也慢，一个更新或是Insert就会很慢，这样看到的就是Update或是Insert怎么这么慢了。所以在数据库写入量或是更新量也比较大的系统，该参数不适合分配过大。而且在高并发，写入量大的系统，建议把该功能禁掉。

query_cache_limit

指定单个查询能够使用的缓冲区大小，缺省为1M

query_cache_min_res_unit

默认是4KB，设置值大对大数据查询有好处，但如果你的查询都是小数据查询，就容易造成内存碎片和浪费



其他缓存怎么开

读缓存，线程缓存，排序缓存

sort_buffer_size = 2M
connection级参数。太大将导致在连接数增高时，内存不足。
max_allowed_packet = 32M
网络传输中一次消息传输量的最大值。系统默认值 为1MB，最大值是1GB，必须设置1024的倍数。
join_buffer_size = 2M
和sort_buffer_size一样，该参数对应的分配内存也是每个连接独享
tmp_table_size = 256M
默认大小是 32M。GROUP BY 多不多的问题
max_heap_table_size = 256M
key_buffer_size = 2048M
索引的缓冲区大小，对于内存在4GB左右的服务器来说，该参数可设置为256MB或384MB。
read_buffer_size = 1M
read_rnd_buffer_size = 16M
进行排序查询时，MySql会首先扫描一遍该缓冲，以避免磁盘搜索
bulk_insert_buffer_size = 64M
批量插入数据缓存大小，可以有效提高插入效率，默认为8M

Innodb缓存

innodb_buffer_pool_size = 2048M
只需要用Innodb的话则可以设置它高达 70-80% 的可用内存。一些应用于 key_buffer 的规则有 — 如果你的数据量不大，并且不会暴增，那么无需把 innodb_buffer_pool_size 设置的太大了。
innodb_additional_mem_pool_size = 16M
网络传输中一次消息传输量的最大值。系统默认值为1MB，最大值是1GB，必须设置1024的倍数。
innodb_log_files_in_group = 3
循环方式将日志文件写到多个文件。推荐设置为3
innodb_lock_wait_timeout = 120
InnoDB 有其内置的死锁检测机制，能导致未完成的事务回滚。innodb_file_per_table = 0
独享表空间，关闭

连接数要多少

连接数

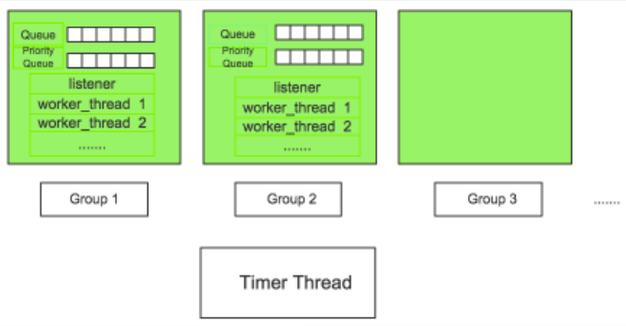
open_files_limit　= 10240
允许打开的文件数
back_log = 600
短时间内的多少个请求可以被存在堆栈中
max_connections = 3000，MySQL默认的最大连接数为100，MySQL服务器允许的最大连接数16384
MySQL允许最大的进程连接数
max_connect_errors = 6000
设置每个主机的连接请求异常中断的最大次数，当超过该次数，MYSQL服务器将禁止host的连接请求
thread_cache_size = 300
重新利用保存在缓存中线程的数量
thread_concurrency = 8
thread_concurrency应设为总CPU核数的2倍
thread_stack = 192K
每个线程的堆栈大小，默认值足够大，可满足普通操作。可设置范围为128K至4GB，默认为192KB。

线程池很少配

线程池有关参数

thread_handling
表示线程池模型。
thread_pool_size
表示线程池的group个数，一般设置为当前CPU核心数目。理想情况下，一个group一个活跃的工作线程，达到充分利用CPU的目的。
thread_pool_stall_limit
用于timer线程定期检查group是否“停滞”，参数表示检测的间隔。
thread_pool_idle_timeout
当一个worker空闲一段时间后会自动退出，保证线程池中的工作线程在满足请求的情况下，保持比较低的水平。60秒
thread_pool_oversubscribe
该参数用于控制CPU核心上“超频”的线程数。这个参数设置值不含listen线程计数。
threadpool_high_prio_mode
表示优先队列的模式。
thread_pool_max_threads
限制线程池最大的线程数，超过将无法再创建更多的线程，默认为100000。
thread_pool_high_prio_tickets
最多语序多少次被放入高优先级队列中，默认为4294967295。
只有在thread_pool_high_prio_mode为transactions的时候才有效果

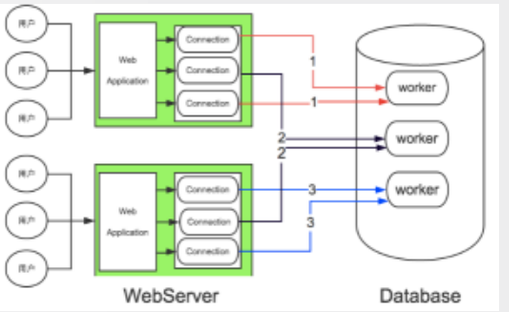
```
#thread pool
thread_handling=pool-of-threads
thread_pool_oversubscribe=3
thread_pool_size=24
performance_schema=off
#extra connection
extra_max_connections = 8
extra_port = 33333
```



每一个绿色的方框代表一个group，group数目由thread_pool_size参数决定。每个group包含一个优先队列和普通队列，包含一个listener线程和若干个工作线程，listener线程和worker线程可以动态转换，worker线程数目由工作负载决定，同时受到thread_pool_oversubscribe设置影响。此外，整个线程池有一个timer线程监控group，防止group“停滞”。

线程处理的最小单位是statement(语句)

线程池实现在server端，通过创建一定数量的线程服务DB请求，相对于one-conection-per-thread的一个线程服务一个连接的方式，线程池服务的最小单位是语句，即一个线程可以对应多个活跃的连接。



慢查询日志

慢查询日志

slow_query_log
是否开启慢查询日志，1表示开启，0表示关闭。
log-slow-queries
旧版（5.6以下版本）MySQL数据库慢查询日志存储路径。可以不设置该参数，系统则会默认给一个缺省的文件host_name-slow.log
slow-query-log-file
新版（5.6及以上版本）MySQL数据库慢查询日志存储路径。可以不设置该参数，系统则会默认给一个缺省的文件host_name-slow.log
long_query_time
慢查询阈值，当查询时间多于设定的阈值时，记录日志。
log_queries_not_using_indexes
未使用索引的查询也被记录到慢查询日志中（可选项）。
log_output
日志存储方式。log_output='FILE'表示将日志存入文件，默认值是'FILE'。log_output='TABLE'表示将日志存入数据库，这样日志信息就会被写入到mysql.slow_log表中。MySQL数据库支持同时两种日志存储方式，配置的时候以逗号隔开即可，如：log_output='FILE,TABLE'。日志记录到系统的专用日志表中，要比记录到文件耗费更多的系统资源，因此对于需要启用慢查询日志，又需要能够获得更高的系统性能，那么建议优先记录到文件。

03

监控工具

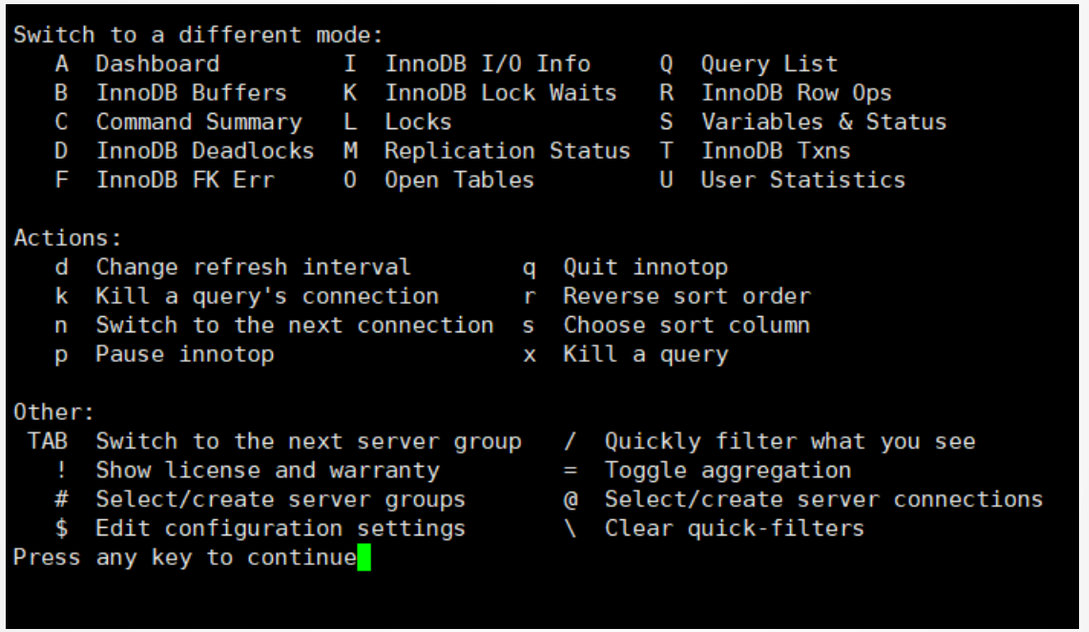
TOOLS

innotop

安装：yum install innotop

启动：innotop -u root -p ‘123’

帮助：?



innotop

查询列表：Q

```
[RO] Query List (? for help)                                                          localhost, 2d22h, 11.43 QPS, 9

When   Load  Cxns   QPS    Slow  Se/In/Up/De%  QCacheHit  KCacheHit  BpsIn  BpsOut
Now    0.01  9      11.43     0  90/ 0/ 0/ 0       0.00%    100.00%  2.18k   3.47k
Total  0.00  2.93k   9.60   219  63/ 3/ 0/ 0       0.00%     40.00%  2.31k  27.35k

Cmd    ID    State         User      Host       DB         Time       Query
Query   1188 Sending data  luokeyua  localhost  hk91coins  00:00.084  select count(*) from fentrustlog_vcoin l, fentrust_vcoin v where l.fen_fid = v.fid and v.fvi_fid = 16 and v.fvi2_fid = 10 and l.fcreatetime < '2018-04-22 09:30:00'
```

缓冲区：B

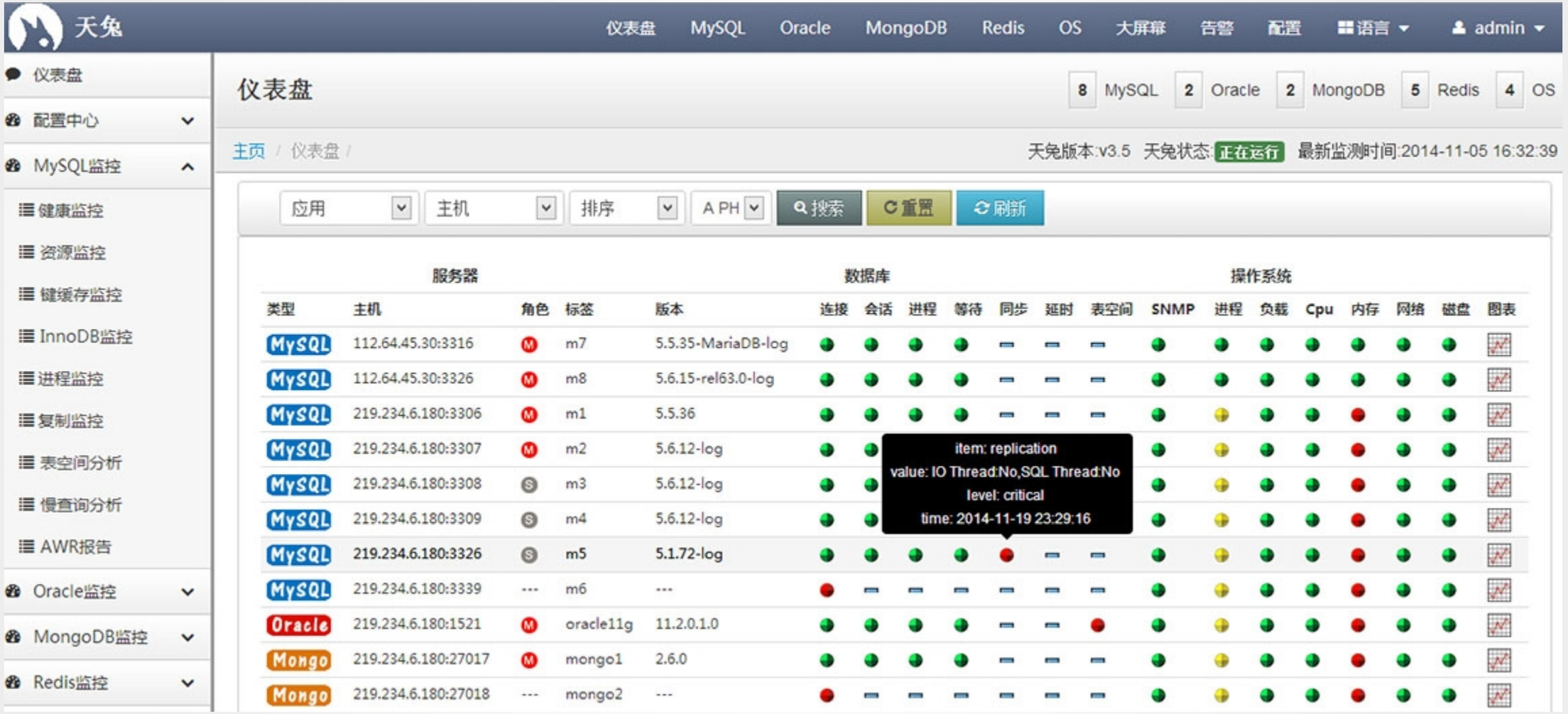
```
[RO] Dashboard (? for help)

Uptime  MaxSQL  ReplLag  QPS    Cxns  Run  Miss   Lock  Tbls  Repl  SQL
 2d22h                   20.75     9    1  76.71     0   400
```

命令统计：C

```
                         Command Summary
Name                        Value    Pct     Last Incr    Pct
Com_select                1554396  63.81%            2  11.76%
Com_set_option             518220  21.27%            6  35.29%
Com_commit                 256271  10.52%            3  17.65%
Com_insert                  88258   3.62%            1   5.88%
Com_update                   9662   0.40%            0   0.00%
Com_show_fields              2822   0.12%            0   0.00%
Com_show_keys                2346   0.10%            0   0.00%
Com_change_db                 911   0.04%            0   0.00%
Com_show_variables            554   0.02%            0   0.00%
Com_show_create_table         421   0.02%            0   0.00%
Com_show_tables               397   0.02%            0   0.00%
Com_show_table_status         339   0.01%            0   0.00%
Com_show_triggers             339   0.01%            0   0.00%
Com_admin_commands            217   0.01%            2  11.76%
Com_rollback                  205   0.01%            0   0.00%
Com_show_status               197   0.01%            2  11.76%
Com_insert_select             151   0.01%            0   0.00%
Com_show_engine_status         36   0.00%            0   0.00%
Com_show_master_status         24   0.00%            1   5.88%
Com_alter_table                22   0.00%            0   0.00%
```

lepus

http://www.lepus.cc/page/product



04

主从复制

MASTER SLAVE

主从复制配置

主从复制配置

log-bin=mysql-bin
[必须]启用二进制日志
server-id=222
[必须]服务器唯一ID，默认是1，一般取IP最后一段
binlog-do-db=DB1
binlog-do-db=DB2 #如果备份多个数据库，重复设置这个选项即可
binlog-do-db=DB3 #需要同步的数据库，如果没有本行，即表示同步所有的数据库
binlog-ignore-db=mysql #被忽略的数据库
rpl_semi_sync_slave_enabled =1 #启用半同步复制
rpl_semi_sync_master_timeout=milliseconds
设置此参数值（ms）,为了防止半同步复制在没有收到确认的情况下发生堵塞，如果Master在超时之前没有收到任何确认，将恢复到正常的异步复制，并继续执行没有半同步的复制操作。
rpl_semi_sync_master_wait_no_slave={ON|OFF}
如果一个事务被提交,但Master没有任何Slave的连接。默认情况下，Master会在时间限制范围内继续等待Slave的连接，并确认该事务已经被正确的写到磁盘上。可以使用此参数选项关闭这种行为，在这种情况下，如果没有Slave连接，Master就会恢复到异步复制。
master
GRANT REPLICATION SLAVE ON *.* to 'mysync'@'%' identified by 'q123456'; //一般不用root帐号，“%”表示所有客户端都可能连，只要帐号，密码正确，此处可用具体客户端IP代替，如192.168.145.226，加强安全。
slave
change master to master_host='192.168.145.222',master_user='mysync',master_password='q123456',
master_log_file='mysql-bin.000004',master_log_pos=308;

主从复制配置

当有一个Slave失去连接后，不影响主库执行操作的时间，建库同样很快；但当所有Slave都关闭后，Master上建库时产生等待，当等待1s后才会执行库建操作；一旦超时，它会自动降级为异步，再执行不会再产生等待。
--此时的客户端已经不存在了

INSTALL PLUGIN rpl_semi_sync_slave SONAME'semisync_slave.so';

Rpl_semi_sync_master_clients
此状态变量报告了支持和注册的半同步复制已连接的Slave数量
Rpl_semi_sync_master_status
Master半同步复制的状态，1是活动状态，0是非活动状态--要么是因为他没有被启用，或是因为它已经恢复到异步复制。
Rpl_semi_sync_slave_status
Slave半同步复制的状态，1是已经被启用，且IO线程正在运行，0是非活动状态

MySQL在加载并开启Semi-sync插件后，每一个事务需等待备库接收日志后才返回给客户端。如果做的是小事务，两台主机的延迟又较小，则Semi-sync可以实现在性能很小损失的情况下的零数据丢失。
所以我们可以做多个备库，任何一个备库接收完成日志后，主库就可以返回给客户端了。

几个忠告

1、拓宽知识面（天文，地理，历史，文化，金融）
2、深入的了解行业知识（IT行业最牛逼的是别人只需学一个专业）
3、逻辑优化同样是优化（新补充的功能考虑普适性）
4、协作还是独行，这是个问题
5、品性的升华，远胜于技能的掌握